Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/MR/7440--00-8246

The Database for the Assessment of Requirements and Tactics (DART) Phase I

JOHN BRECKENRIDGE
FRANK MCCREEDY
KEVIN SHAW
RALPH PERNICIARO

Mapping, Charting, & Geodesy Branch
Marine Geosciences Division

September 29, 2000

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 29, 2000	Memorandum Report

4. TITLE AND SUBTITLE

The Database for the Assessment of Requirements and Tactics (DART) Phase I

5. FUNDING NUMBERS

TA-OMANG

6. AUTHOR(S)

John Breckenridge, Frank McCreedy, Kevin Shaw, and Ralph Perniciaro

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/7440--00-8246

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Guard Bureau-Counterdrug Directorate
Georgia Tech Research Institute
Baker Building on Dalney Road, Room 241B
Atlanta, GA 30332-0841

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

The DART establishes an online method of examining the current information technology (IT) resources of law enforcement agents (LEA) throughout the United States. The DART interviewing tool uses the World Wide Web as a delivery vehicle for conducting a direct census of LEA involved in counter-drug operations. By establishing a baseline understanding of current LEA IT resources, the DART can assist in determining nationwide resource requirements for the deployment of the CD-Geographic Regional Assessment Sensor System (CD-GRASS). Deficiencies in local LEA resources can be identified and addressed by the regional CounterDrug Coordinator (CDC) in an appropriate time frame to avoid significant impacts upon the deployment of the CD-GRASS.

14. SUBJECT TERMS

15. NUMBER OF PAGES

156

16. PRICE CODE

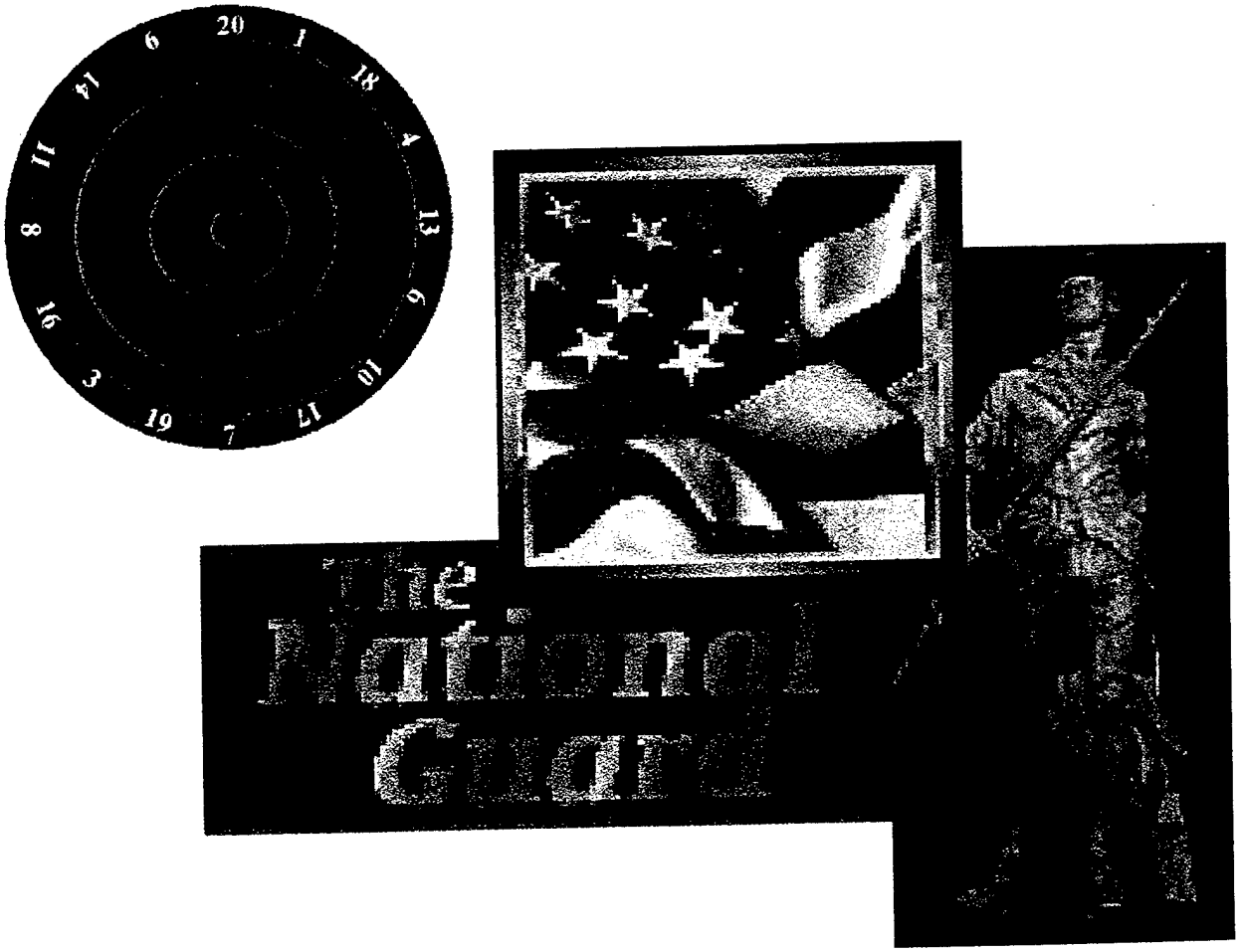
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

The Database for the Assessment of Requirements and Tactics (DART)

Phase I: Development of a World Wide Web-based Information Technology (IT) Census



Prepared for the National Guard Bureau – Counterdrug Directorate
By Naval Research Laboratory, Code 7440.2, Stennis Space Center, MS

Table of Contents

1.0 Abstract: The National Guard Bureau – Counterdrug Directorate (NGB-CD) is developing the Counterdrug Geographical Regional Assessment Sensor System (CD-GRASS) as a nationwide force multiplier to provide Law Enforcement Agents (LEA) with direct application support in applying the latest spatial data technology. The mission of CD-GRASS, as defined by Higgins, et.al. (2000) focuses on information integration, assessment, and decision support to counterdrug operations through state-of-the-art technology transfer and insertion. The deployment of CD-GRASS is directly dependent upon an adequate understanding of the Information Technology (IT) resources available to the LEA community. This report addresses the development of a Database for the Assessment of Requirements and Tactics (DART) as a means of understanding the current availability of IT resources among LEA. It describes the development of a JAVA™ and ArcView™ based technical approach to executing a nationwide digital census of LEA. The information will be used to describe LEA's ability to participate in the CD-GRASS from an IT perspective.

2.0 Introduction: The DART establishes an online database of current IT resources for LEA throughout the United States. The DART interviewing tool uses the World Wide Web as a delivery vehicle for conducting a direct census of LEA involved in counter-drug operations. By establishing a baseline understanding of current LEA IT resources, the DART can assist in determining nationwide resource requirements for the deployment of the CD-GRASS. Deficiencies in local LEA resources can be identified and addressed by the regional CounterDrug Coordinator (CDC) in an appropriate timeframe to avoid significant impacts upon the deployment of the CD-GRASS.

3.0 Purpose: The development of the DART as a Web-based graphical user interface (GUI) provides a means of conducting an online interactive IT Census of the counter-drug LEA community. Although, it relies upon the technical disciplines of Survey Research and Geographic Information Systems (GIS), its primary goal is to collect and report a baseline of information concerning LEA IT resources.

4.0 Survey Approach: Survey Research includes the use of survey sampling and statistical analysis as the primary steps in developing a concise and accurate understanding of a specialized population. An advanced study of the LEA population (i.e. survey research) would rely heavily upon various aspects of statistics, sampling and testing theory to isolate and extract information from the entire LEA population or individual cross-sections (i.e. states). Weinberg et. al. (1981) remind us that the two main approaches of statistical analysis include descriptive and sampling statistics. In survey research, these translate respectively to functional approaches of 1) conducting a full census of an entire population, or 2) gathering an appropriate representative sample of the population to support statistical inference about the entire population (Ross et al., 1993). Ross et. al. (1993) further describes the general survey process as composing the following steps:

1. Define a **primary purpose for the survey**;
2. State a significant **main objective or hypothesis** that translates the purpose into a verifiable research initiative;
3. Develop a **survey plan** inclusive of a data collection plan, data reduction/reformatting plan, and an analysis plan.

Manuscript approved August 9, 2000.

NRL will follow this outline in carrying out the LEA IT Census. The following sections further examine specific considerations relative to that census.

4.1 Census vs. Sampling: A significant question must be raised in establishing a plan for surveying a population such as the CD-LEA. Should the survey involve a full census of each member, or can a smaller sample group be used to infer an accurate understanding of all LEA IT resources? In most cases the answer depends upon the size of the study population and the reliability and validity of the survey instrument. However, another factor is whether the information about individual members of the population will actually be used in its basic format, or will it be reduced and/or grouped to offer a condensed assessment of the total population. In the case of the DART IT Census, the long term use of the data needs to support the CDC's examination of both individual LEA and the total LEA population of their state. The CDC will examine individual LEA to offer them assistance in meeting the IT requirements of the CD-GRASS. State-level reports will be used to offer a collective view of all LEA's capabilities and long term requirements for the state jurisdiction. Therefore it was concluded that a complete survey of the LEA population would benefit the deployment of CD-GRASS the most.

4.2 Variance: The statistical variance of individual characteristics within a census population can vary due to differences in either the individual members themselves or in the parameters used to measure that population. As defined by Weinburg et al. (1981), variance refers to the relationship among all terms (i.e. observed members) in a distribution (i.e. population) considered together. This definition focuses more on how the characteristics of each member relate to other members of the same population. However, if an ideal level of goodness or quality (i.e. baseline set of requirements) is applied as a model for any member, than the variability between members becomes a measure of their ability to fit within an ideal population. In fact, the significance or insignificance of this variability among the individual members can be interpreted as a quantitative scale of each member's ability to function in that population. The measure of variability can then be directly correlated with the survey purpose and objective. Therefore, the objective of most surveys is to prove either 1) how similar or different the members are within a given population, or 2) how alike or different individuals of a population are in comparison to its most desirable condition. It is the latter case that most precisely fits the purpose of the DART IT Survey. The DART IT Survey will determine if individual LEA are able to participate as a functional member of CD-GRASS users community.

4.3 Geographic Information Systems (GIS): GIS provide a technical approach to managing spatially referenced data to produce information that helps a user accomplish a specialized mission, (i.e. CounterDrug Operations). The term "GIS" is used interchangeably to denote aspects of either a scientific discipline and/or various computer systems that manage digital information based upon its geographic characteristics. Consequently, the IT Census asks spatially related questions about LEA: 1) Who is the LEA, and 2) What IT resources does he/she have available to assist in carrying out their counter-drug mission, and 3) Where are these individual resources located in relation to the entire state LEA population? The use of GIS in conjunction with these questions will allow the NGB to examine spatial distributions of LEA and their IT resources on both a national and regional basis.

The components (i.e. subsystems) of a GIS support the basic functions of 1) data input, 2) data storage and retrieval, 3) data manipulation/analysis, and 4) data reporting/application (Marble, et. al. 1984). As a scientific process, these functions facilitate the integration of five types of resources that collectively comprise the operational environment of any GIS: 1) people, 2) data, 3) hardware, 4) software, and 5) information products. The order of importance of these entities will vary with the user's perspective of GIS. It is important to understand, however, that a GIS does not function without interaction of people tailored around a specific mission. The design of the DART IT Census is no exception. The DART IT Census supports the CDC in maintaining an ongoing understanding of LEA IT resources within a given jurisdiction.

5.0 Detailed Process of the DART IT Census: The following sections present a survey plan for conducting an online interactive nationwide census of LEA and their IT resources. By following this approach, NGB-CD will have the information needed to support the determination of individual and collective IT capabilities at the local LEA level in relation to impacts upon the CD-GRASS.

In the case of the LEA IT Census, the size of the total population is unknown although each LEA can be associated with one of 54 CDC. Since each LEA can eventually be isolated within each CDC's jurisdiction, the total population of LEA is considered to be of a size reasonable enough that it can be counted in its entirety. The use of 54 CDC offers an ideal set of intermediate control points to the LEA population. For this reason, the use of descriptive statistics (i.e. census) is considered the most effective means of measuring all individual LEA's IT resources.

5.1 Survey Purpose: Determine the IT capabilities of CounterDrug Law Enforcement Agents nationwide.

5.2 Survey Objective: Establish a queriable database of LEA IT assets and connectivity.

5.3 Survey Plan: DART will apply quantitative statistical analysis to determine if individual LEA IT resources meet a baseline CD-GRASS criteria that requires:

- One standalone personal computer
- Minimum computer operating speed of 200 MHz
- Operating System (OS) compatibility to Windows 95/98/2000
- Direct network connection
- Common World Wide Web browser interface.

These criteria can be divided into four primary categories of information: Hardware, Software/OS Compatibility, Network Connectivity, and World Wide Web Interface. Collectively, they represent 100% of the ideal user model for LEA IT functionality in the CD-GRASS. Although the normal distribution of LEA will include members that greatly exceed these capabilities, all LEA participating in CD-GRASS will be expected to meet these

minimum IT requirements. These categories can also be used to define an outline for our survey questionnaire and serve well to direct our statistical analysis.

Population of the IT Census will require that a formal Census Campaign Period (CCP) be declared by NGB-CD to assist CDC in soliciting LEA involvement in the information collection process. This CCP can be initiated through a combination of CDC State level visits, DART presentations at Law Enforcement Conferences, and mail out solicitations from NGB-CD. Regardless of the method used, a primary incentive for LEA to participate in the census should be communicated during this timeframe. Obviously, LEA's future ability to utilize CD-GRASS could be one such incentive. Although this would involve an education process as to what the benefits of CD-GRASS will be for the LEA. The CCP may need to be conducted on an annual or even ongoing basis that assures the census of new LEA. LEA with no IT resources would be contacted by the CDC, who in turn would enter their information in the DART system.

5.4 **Survey Technical Approach:** Based upon the success of similar NRL efforts to collect digital spatial data requirements, a technical approach was chosen to develop an interactive (i.e. online) GIS-linked questionnaire. Using JAVA 1.1 and ArcView 3.2, NRL designed the DART IT Census by developing the following components:

- A JAVA-based Applet that accepts IT resource information and populates an ODBC database
- An ArcView GIS-based map interface that presents the CDC as the primary point of contact for LEA in each state or commonwealth
- A GIS tutorial function that provides a basic function of educating the LEA on spatial data concepts
- Online documentation of the DART Administrative Program documentation including briefing materials and formal publications.

5.5 **Data Reduction and Reformatting:** This function of the survey serves to eliminate unnecessary data while also transforming questionnaire answers into statistically usable data. Unlike manual methods that have a strong potential for error and can be tedious and labor intensive (Ross, et. al., 1993), digital surveys offer the advantage that data can be categorized, reduced and reformatted at the time of collection. For the purposes of the IT Census, minimal data reduction will take place. The survey uses a small number of key factors to identify whether an LEA's IT assets meet the minimal standard identified in the survey plan. Therefore, primary data reduction was limited to reformatting all textual answers into numerical values stored in a primary database field called **Cummulative_val**. Again, the number of parameters involved allowed both the original text value and reduced data to easily be stored in the database.

5.6 **Quantitative Analysis:** Since the purpose of the IT Census is primarily to report only the resource holdings of the LEA, very limited statistical capabilities are required of the DART GUI. However, the added advantage of having both GIS and an ODBC database framework for the IT Census is that statistical computing functions are readily available if desired at a later time. Thus the focus of the quantitative analysis required for the IT Census is two fold

and includes: 1) adding a reporting function that allows the CDC to readily review individual LEA response, and 2) storing LEA responses in a quantitative format that facilitates the add-on of statistical functions.

In reviewing the five baseline IT criteria for CD-GRASS, we recall that each LEA is required to meet an operational IT baseline. Since each of these baseline criteria can be logically grouped into four major categories of 1) hardware, 2) software/operating system, 3) network connection, and 4) Web access, numeric values of 1 or 0 can be assessed to each question in these categories. This provides a quantitative measurement for each LEA response. The data for each question is stored as a simple binary value, that is 1 is stored for positive answers (e.g. yes, descriptive values), 0 for negative answers (e.g. no, none, etc.). In doing so, the primary statistical function of reporting a LEA's ability to meet the baseline criteria can be easily accomplished. In fact, the query of all LEA who answer "no" to the first question (i.e. Do you have a computer) is a direct selection of LEA who do not meet the baseline requirements. Additional consideration must be given to those who do have equipment but also have inadequate access to the network or a Web browser. As a binary format the LEA answers are stored in a numerical primary field called "questions_num" and can have a cumulatively sum of '10' possible points. Obviously those values significantly less than '10' represent LEA that do not meet the baseline requirements.

The manner in which the LEA answers are stored also makes it possible to enhance the statistics at a later data. Two other methods of representing the values involve assigning an index of weighted values to prioritize the importance of each category of answers. These approaches include 1) assigning a significant value that is multiplied by each answer value or the sum of values in the four major categories, and 2) setting a data schema that uses the significant digits of the "cumulative answer" field to represent their importance. In the first approach, making the cumulative score of factors 1 and 2 a high enough percentage of the total (e.g., 75 of 100 total points) is one method of ensuring that their effect on the score is obvious during any statistical operations. Alternatively, the second approach would use significant decimal places in a cumulative score to represent significance in each of the four main categories, thus the hundreds and thousands place of a 4-digit real number (i.e. 5025) would correlate to the most significant categories of the questions. Using the thousands place to represent the hardware category, any LEA with a score in the 4000 to 5000 range might be considered compliant with the CD-GRASS baseline for hardware. Values within the 3000 range would be marginally suitable, while 1000-2000 range values would represent significant deficiency upon the LEA to meet the baseline requirements. With either method, the value that results from the scoring of factors within one significant category can be readily distinguishable in comparison to the factors in other categories. In both cases the independent variables can receive much greater values than those of the dependent variables, making it possible to alter the focus of any statistical analysis functions. Likewise, analysis tools can be added to measure the variability of the LEA's answers in relation to either the total population or a subset (i.e. CONUS vs. statewide analysis). Since these scalar values can be applied at either the time of data reduction or later as a weighted index, expanding the statistical functionality is not limited by this design phase of the IT Census.

6.0 Problems Encountered: Technical development of the DART IT Census was relatively problem free in relation to the task of establishing a JAVA-based Web design. This was

primarily due to lessons learned in previous NRL research efforts. The reduced start-up time allowed NRL to concentrate on three major areas: 1) the functionality of the user relative to the questionnaire, 2) statistical evaluation of the data, and 3) the GIS map interface. Significant problems encountered in the later area required a serious look at the functionality of supporting a Client/Server mode of operation for the spatial data aspects of the DART IT Census.

6.1 Client Server Lockups Due to Collision of Simultaneous Users: During a sponsor review of the DART GUI, Mr. Nick Faust, GTRI discovered a serious conflict resulting from the collision of multiple user requests submitted simultaneously to the ArcView® Server via Remote Procedure Calls (RPC). In fact, lockup of the ArcView Server could be anticipated and purposefully re-created when more than one user simultaneously accessed the DART JAVA Applet. The problem was specifically related to features that generated calls to ArcView to create a feature display file (e.g. jpg file) as a result of an RPC request through the ArcView Server. One possible cause for this conflict may be the manner in which the RPC code and/or its associated map display code may be written to expect a return response to the server once a user request is completed.

6.2 Solutions: Avenue® code was written to set up a lock/unlock mechanism to avoid simultaneity. This code either did not work correctly or the problem was occurring before that point in the RPC code or in the ArcView InterApplication Communicator ™ (i.e. iac.exe). IAC, as defined by the ArcView Online Help, generically refers to the ability to have two or more applications communicate (ESRI, 1990-1998). In contrast, the RPC is carried out by an executable (i.e. iac.exe) that is called from a Java servlet. It was apparent to our programmers that the most consistent manner of avoiding simultaneous processing in ArcView would be to regulate when the iac.exe is called from the servlet. Thus the ArcView server would continue to operate in its default manner of processing individual requests. One advantage to addressing the problem from this level was that it could be done within the servlet code, both eliminating a need for more in-depth understanding of the IAC server interface and allowing the programmers to work in a open programming environment that they had constructed.

In general, servlets can be implemented through two primary methods:

1. SingleThreadModel: Each client gets its own instance of the servlet. No method is called simultaneously since each instance has its own copy of the method. A second instance call of a same method may perform the same function, but it can be thought of as a separate method since every variable it acts upon is separate from the variables in the other instances.

2. Default: Each client gets a thread that makes calls into one instance of the servlet allowing a method to be called simultaneously.

The solution developed by NRL uses the SingleThreadModel. It works when a client makes a user request by having the servlet check a flag (i.e. lock) to see if another instance of the servlet is already calling the iac.exe. This is possible since each client receives its own instance of the servlet under the current scheme using the SingleThreadModel. If the flag is

true, the servlet waits 100 milliseconds and tries again until it is able to set the flag (i.e. lock the lock). After it has finished its call through the servlet, the flag is unset (i.e. unlocks the lock). Since each system resource has an active state that controls its ability for sharing, the user can check this state condition. By keeping an active check on the state of user requests, we are then able to use a SingleThreadModel that allows us to keep resources (i.e.database connections, etc) from being shared in a conflicting way. This approach eliminates the possibility that the state of requests passed through the servlet can become undeterminable. The solution insures that all of the resources can be individually locked and unlocked so as to avoid any simultaneous attempt to execute on the server side.

In examining the functional results of using the SingleThreadModel, NRL found it eliminated the server lockup problems previously encountered. The solution also did not seem to introduce any appreciable differences in response times from the ArcView Server. In some cases, it appeared that the more intensive requests for query might have been improved some as a result of the implementation of this servlet code.

7.0 Accessing the DART JAVA Applet: The DART JAVA Applet is designed to allow access via a common web browser, (e.g., Netscape 4.6 or MS Internet Explorer) with JAVA Applet functions enabled. Currently the World Wide Web page for the DART Applet can be accessed at http://redwood.nrlssc.navy.mil/DartApplet/DartApplet.html. Illustration 1:DART Conceptual Design offers a high-level flow diagram of the general functionality available through the DART Applet.

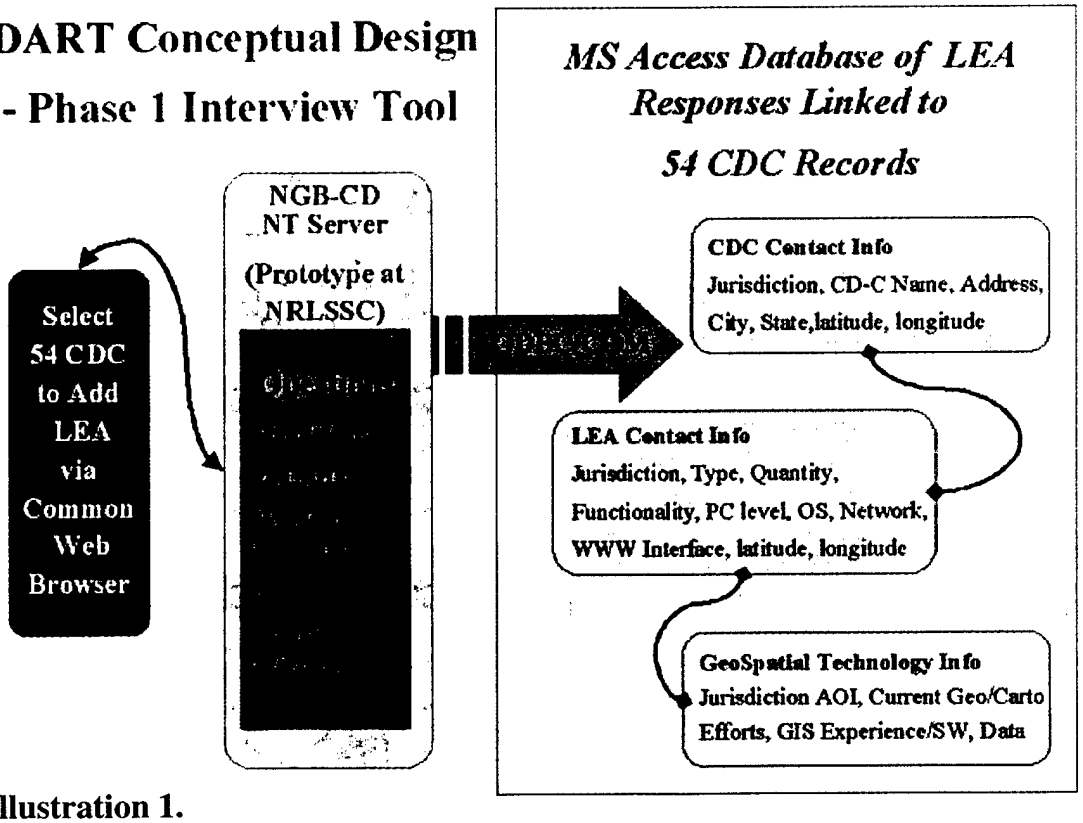


Illustration 1.

The DART Applet contains several information resources that offer the user general information about the DART program and the functional aspects of the DART Applet. The Appendices of this document offer illustrations of several of these information sources. Included in this text are Appendix A) that offers a graphical presentation of the computer screen displays as seen by a DART CDC type user. Appendix B) provides a graphical presentation of the computer screen displays as seen by a DART LEA type user. Appendix C) offers a presentation of the vugraphs describing the DART research effort. These slides can be accessed online in the DART GUI by clicking on the hypertext "DART" on the title screen of the DART JAVA Applet. Appendix D) gives a graphical illustration of the DART online GIS Tutorial and its ArcView™ Remote Procedure Call (RPC) based functionality. Finally, Appendix D) presents the user with a hard copy printout of all JAVA™ and ArcView Avenue™ source code developed in the course of this project. Digital copy of this code and all data, software extensions, and parameter settings required for installation of the DART are also available on a limited basis. For more information concerning this resources contact John Breckenridge, NRL Code 7440.2, Stennis Space Center, MS, 39529 or email: jbreck@nrlssc.navy.mil.

8.0 Conclusion: The DART IT Census has been successfully developed as an online, Web-based tool for conducting a census of CD LEA IT resources on a national basis. Through use of JAVA, ArcView, and ODBC technology, LEA can access the DART WebPages to enter and edit information concerning IT resource capabilities. Likewise, the CDC can utilize the DART to assist in assessing regional impacts upon LEA's ability to actively participate in the upcoming CD-GRASS.

9.0 Acknowledgements: The National Guard Bureau – Counterdrug Directorate (NGB-CD) has funded the development of the Database for the Assessment of Requirements and Tactics (DART) under NGB Primary Source Code "OMANG." The NGB-CD will use DART to support its role of providing enhanced resources to Law Enforcement Agents (LEA) and serving as a force multiplier for civil authorities in counterdrug operations. NRL acknowledges the invaluable commitment and support provided by Lt Col George W. "Billy" Asbell throughout the course of the DART program. Likewise the technical assistance provided by Dr. Melinda Higgins and Mr. Nickolas Faust, Georgia Tech. Research Institute and LCDR Paul Thorpe, USNR helped to ensure a beneficial test and evaluation phase for the DART software and documentation.

10.0 References: The following sections offer bibliographic references, acronym definitions, and a distribution list for this report.

10.1 Bibliography:

Dale, A., Arber, S., Procter, M. (1988). Doing Secondary Analysis, Unwin Hyman, London, England.

Environmental Sciences Research Institute (1990-1998). ArcView HELP, ESRI, Redlands, CA.

Environmental Sciences Research Institute (2000), GIS.com Webpage. http://www.gis.com, ESRI, Redlands, CA.

Fink, A., Kosecoff, J. (1998). How to Conduct Surveys, A Step-By-Step Guide, 2nd Edition, Sage Publications, Thousand Oaks, CA.

GeoComm International Corporation (2000), GeoData Depot Webpage. http://www.gisdatadepot .com/, GeoComm I.C., Niceville, Florida

GISLinx (2000), GISLinx™ Webpage. http://www.gislinx.com/Miscellaneous /Links/index.shtml , Wylie, Texas, 75098, USA.

Higgins, M., Faust, N., and Asbell, W., (2000), Information Insert to GIS Technologies Symposium, Held April 27, 2000, NGB-CD, Washington, D.C.

Krewski, D., Platek, R., Rao, J. N. K. (1981). Current Topics in Survey Sampling, Academic Press, Inc., New York, New York.

Marble, D. F., Calkins, H. W., and Peuguet, D. J. (1984). Basic Readings in Geographic Information Systems, SPAD Systems, LTD., Williamsville, NY.

Ross, K. C., Clark, L. D., Padgett, T. C. (1993). Air University Sampling and Survey Handbook, Air University, Maxwell AFB, AL.

Rossi, P. H., Wright, J. D., Anderson, A. B. (1989). Handbook of Survey Research, Academic Press, Inc., San Diego, CA.

Weinberg, G. H., Schumaker, J. A., and Oltman, D. (1981). Statistics: An Intuitive Approach, 4th ed., Brooks/Cole Publishing Co., Monterey, CA.

10.1 Illustrations:

Illustration 1 DART Conceptual Design

10.2 List of Acronyms

CD-GRASS	Counterdrug Geographical Regional Assessment Sensor System
CDC	CounterDrug Coordinator
CONUS	Continental United States
GIS	Geographic Information Systems
GUI	Graphical User Interface
IT	Information Technology
LEA	Law Enforcement Agents
MHz	Megahertz

NGB-CD	National Guard Bureau – Counterdrug Directorate
ODBC	Open Data Base Connectivity
RPC	Remote Procedure Call

10.3 Distribution List:

Lt Col George W. (Billy) Asbell (8)
National Guard Bureau-Counterdrug Directorate
GTRI EOEML
241B Baker Building
Atlanta, GA 30332-0841

Mr. Nicolas Faust (1)
Dr. Melinda Higgins (1)
Georgia Tech Research Institute
241B Baker Building
Atlanta, GA 30332-0841

NRL Code 7440.2 (5)
John Breckenridge

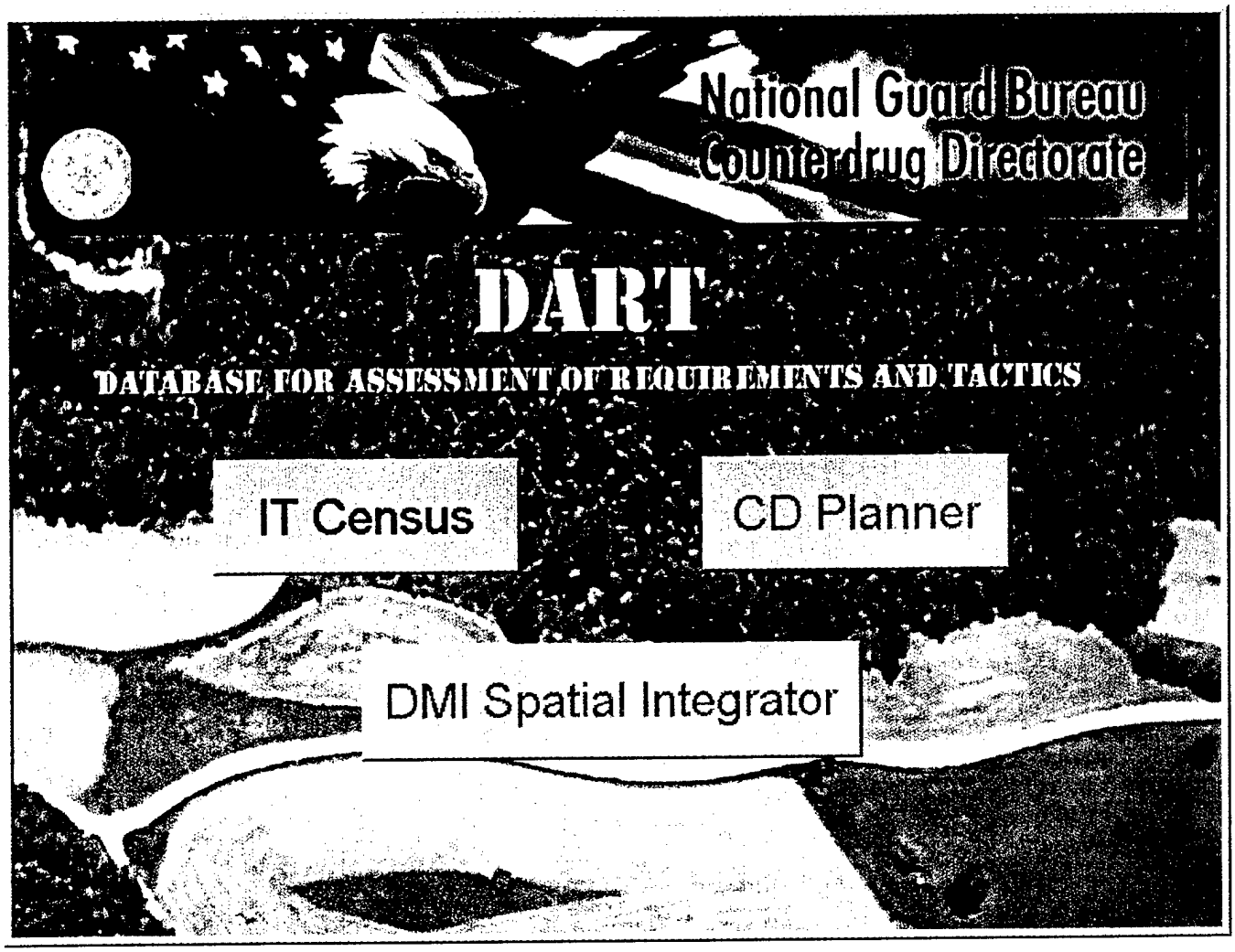
The Database for the Assessment of Requirements and Tactics (DART)

Appendix A: DART WebPages Interface for the CounterDrug Coordinator

Approved for Public Release; distribution is unlimited.

Appendix A. DART WebPages Interface for CounterDrug Coordinator

Appendix A offers a graphical presentation of the screens displayed to the DART Counter-Drug Coordinator (CDC) type user. As the CDC logs into the DART, he/she specifies the state in which they serve as CDC. The following screens present the CDC with the options to view and edit his/her own database information or any information stored about Law Enforcement Agents within their jurisdiction. The following pages also present the typical screen responses requested of the CDC when executing DART software functions (e.g. submit edits command).



Login

Please select your type

of counter-drug position:

☑ CDC ☐ LEA

User Name: jhills

Password: **********

OK Cancel

CDC SELECTION - Select region and state

CDC Info | LEA Info | Hardware | OS/Software | Network | Web Connect

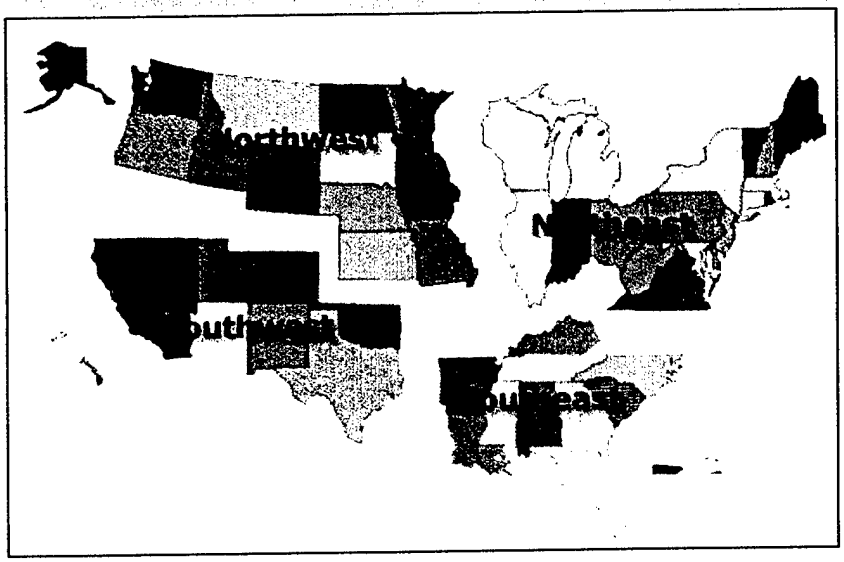
Selected Region: Southeast

State/Territory:

MS
NC
PR
SC
TN

CDC:

JAMES "PARKER" HILLS



Back | Next Page | Home

CDC Info

CDC Info | LEA Info | Hardware | OS/Software | Network | Web Connect

State:

MS
NC
PR
SC
TN

Current CD Coordinator:

JAMES "PARKER" HILLS

CDC Jurisdiction:

ARNG

Address:

NGMS-OTO-DS, 141 MILITARY DR.

Edit

City:

JACKSON

State:

MS

Zip:

39208-8860

Rank/Title:

COL

Comm. Voice:

(601) 313-167

Comm. Fax:

(601) 313-167

DSN, Voice:

293-1670

DSN, Fax:

293-1673

Email:

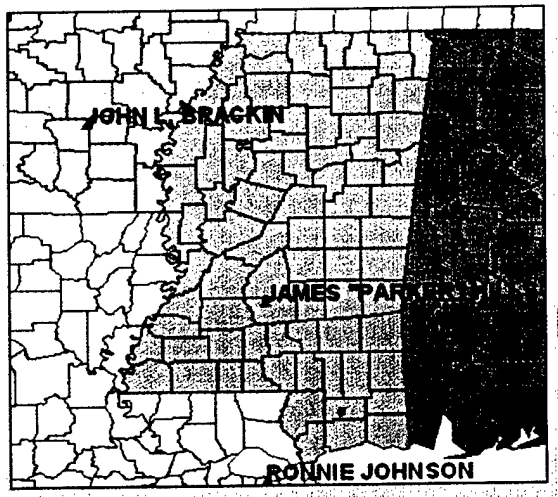
URL:

Username:

JHILLS

Password:

JACKSONMS



Lat:

32.0

Lon:

-90.0

Back | Next Page | Home

LEA Info for CDC JAMES "PARKER" HILLS

CDC Info | LEA Info | Hardware | OS/Software | Network | Web Connect

LEAs:
Herewe Goag
Herewego Aga
john breckenri
John Breckenr
John Smith
Kevin Shaw
Plavit AgainSa

LEA Name: John Smith

Jurisdiction Type: County

Jurisdiction Name: Jackson

City: Pascagoula

State: MS

Zip: 39455

Rank/Title: Col

Comm. Voice: 444-444-4444

Comm. Fax: 555-5555

DSN, Voice: 55

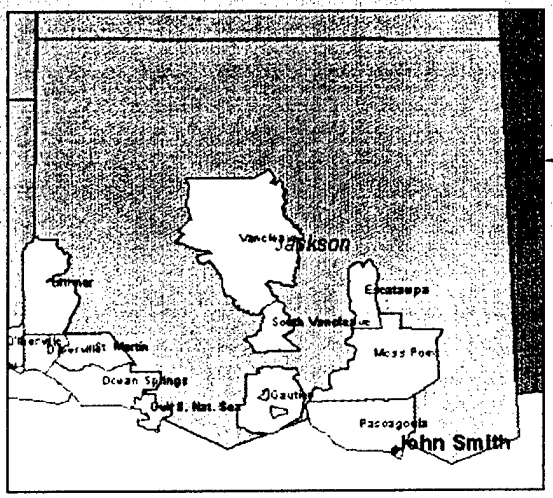
DSN, Fax: 55

Email: js@jackson.go

URL: http://www.yahc

Address: 144 Dover Drive

Edit



Overview

Lat: 30.34

Lon: -88.52

Back | Next Page | Home | Census Report

LEA Info for CDC JAMES "PARKER" HILLS

CDC Info | LEA Info | Hardware | OS/Software | Network | Web Connect

LEAs:
Herewe Goag
Herewego Aga
john breckenri
John Breckenr
John Smith
Kevin Shaw
Plavit AgainSa

LEA Name: John Smith

Jurisdiction Typ: County

Jurisdiction Narr: Jackson

City: Pascagoula

State: MS

Zip: 39455

Rank/Title: Col

Comm. Voice: 444-444-4444

Comm. Fax: 555-5555

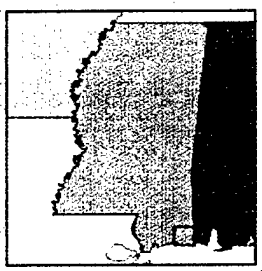
DSN, Voice: 55

DSN, Fax: 55

Email: js@jackson.go

URL: http://www.yahc

Map Overview



OK

Unsigned Java Applet Window

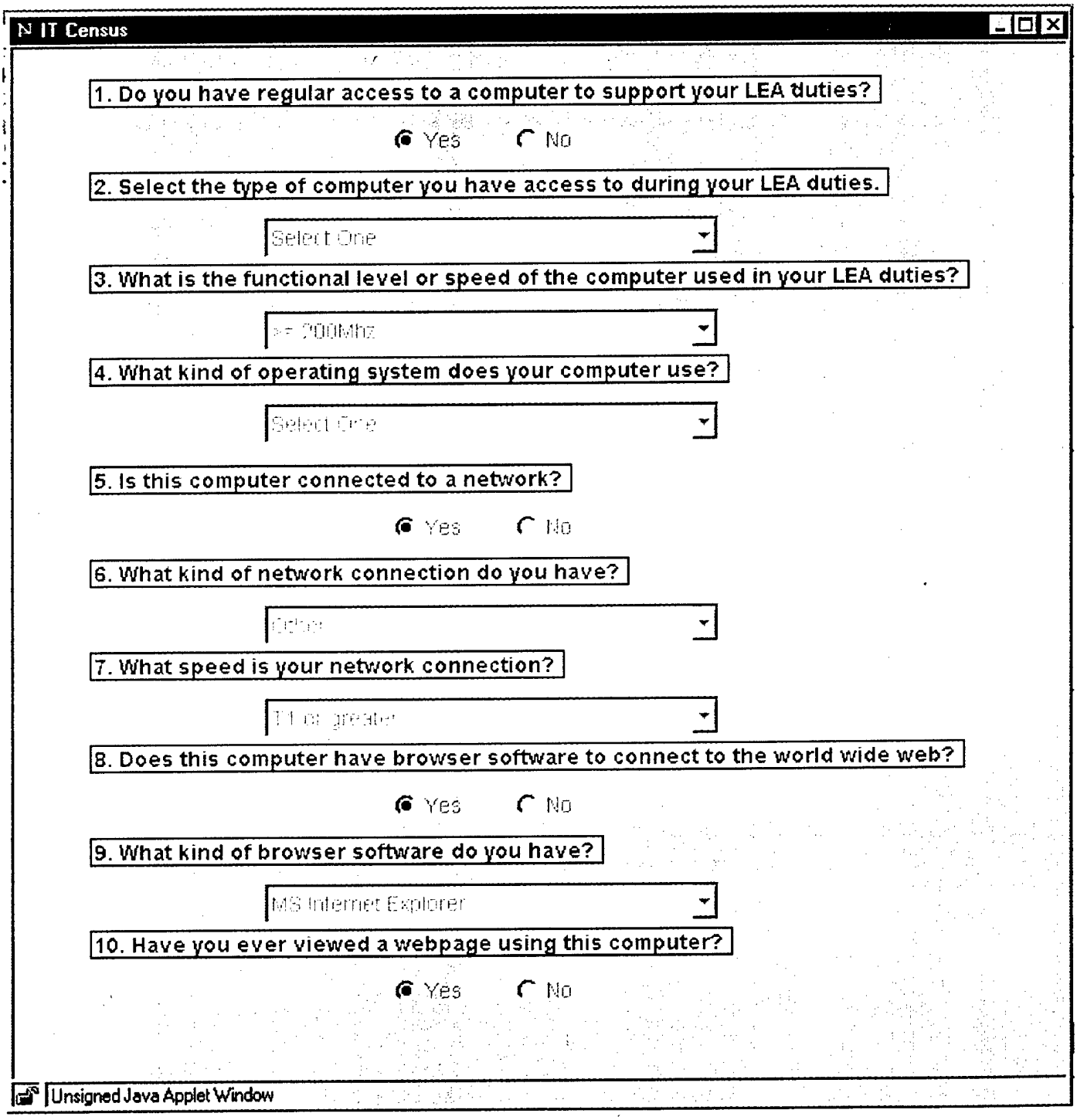
Edit

Overview

Lat: 30.34

Lon: -88.52

Back | Next Page | Home | Census Report


IT Census
1. Do you have regular access to a computer to support your LEA duties?
Yes
No
2. Select the type of computer you have access to during your LEA duties.
Select One
3. What is the functional level or speed of the computer used in your LEA duties?
>= 700Mhz
4. What kind of operating system does your computer use?
Select One
5. Is this computer connected to a network?
Yes
No
6. What kind of network connection do you have?
Other
7. What speed is your network connection?
T1 or greater
8. Does this computer have browser software to connect to the world wide web?
Yes
No
9. What kind of browser software do you have?
MS Internet Explorer
10. Have you ever viewed a webpage using this computer?
Yes
No
Unsigned Java Applet Window

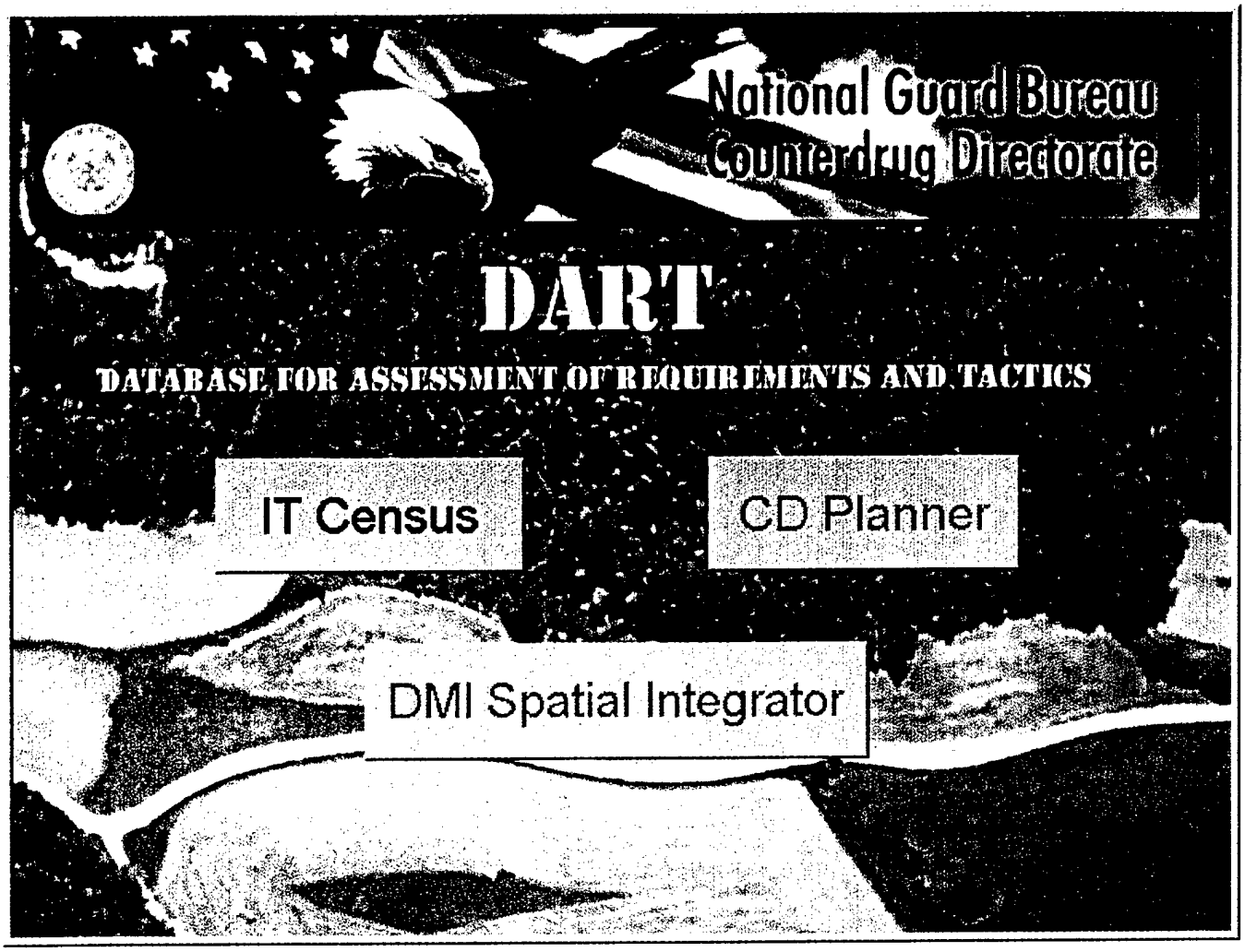
The Database for the Assessment of Requirements and Tactics (DART)

Appendix B: DART WebPages Interface for Law Enforcement Agents

Approved for Public Release; distribution is unlimited.

Appendix B. DART WebPages Interface for Law Enforcement Agents

Appendix B offers a graphical presentation of the screens displayed to the DART Law Enforcement Agents (LEA) type user. As the LEA logs into the DART, he/she specifies the state in which they serve and thus the CDC for that jurisdiction is also identified from the database. The following screens present the LEA with the options to view and edit his/her own database information or view any information stored about the CDC or any other LEA within that jurisdiction. The following pages also present the typical screen responses requested of the LEA as they execute DART software functions (e.g. submit edits command).



Login

Please select your type of counter-drug position:

☐ CDC ☑ LEA

Have you ever logged in before?

Yes | No

New LEA

Please follow the instructions in the popup dialog box on each of the screens. The red arrow indicates the next question you should answer.

Thank you for your participation.

Begin IT Census

Directions

Use the map to select your region of operations. Then select your state from the List. Select "Next Page" when you are ready to continue.

OK

Either select the "Enter" key, "Tab" key or Click on a question to continue.

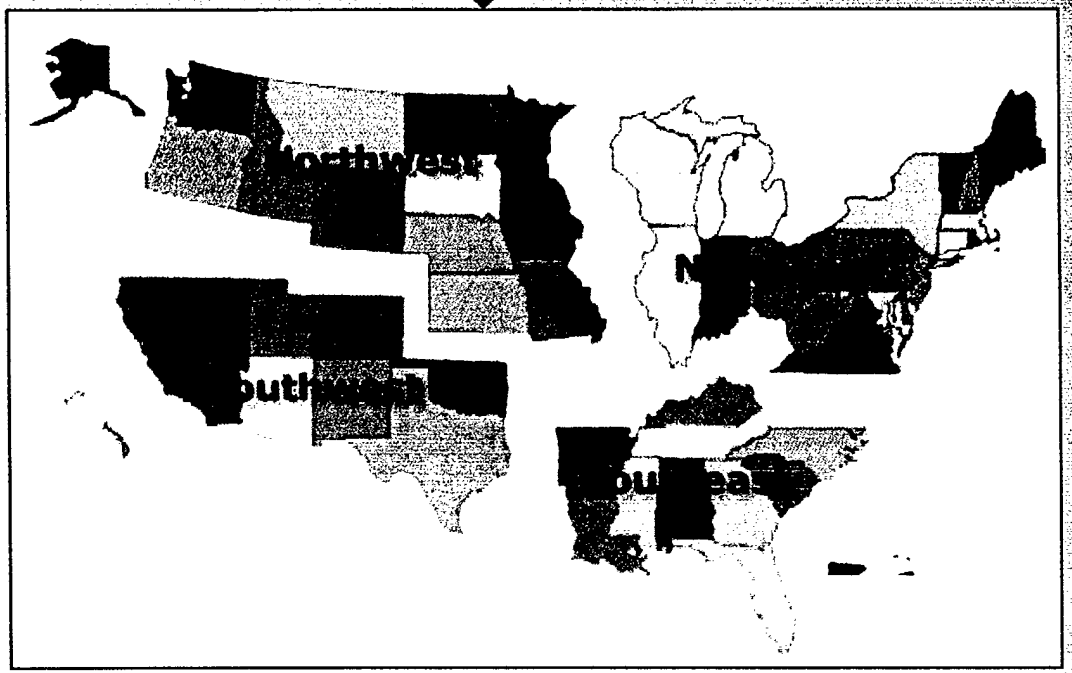
Unsigned Java Applet Window

navy.mil/DartApplet/DartApplet.htm

Netscape Print Security Stop

ELECTION - Select region and state

EA Info | Hardware | OS/Software | Network | Web Connect



State/Territory:

CDC:

Back | Next Page | Home

CDC Info

CDC Info | LEA Info | Hardware | OS/Software | Network | Web Connect

State: MS NC

Current CD Coordinator: JAMES "PARKER" HILLS

Address: NGMS-OTO-DS, 141 MILITARY DR.

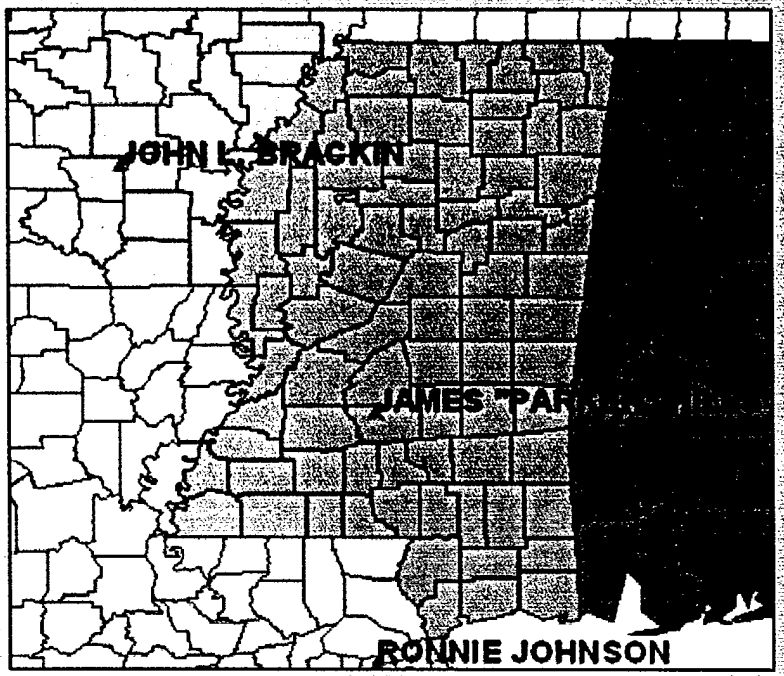
Directions

This is the information for the state you have selected and the Counter-Drug Coordinator (CDC) for your region. If this information appears correct, press "Next Page" to continue. If not, select another state from the

OK

Either select the "Enter" key, "Tab" key or Click on a question to continue.

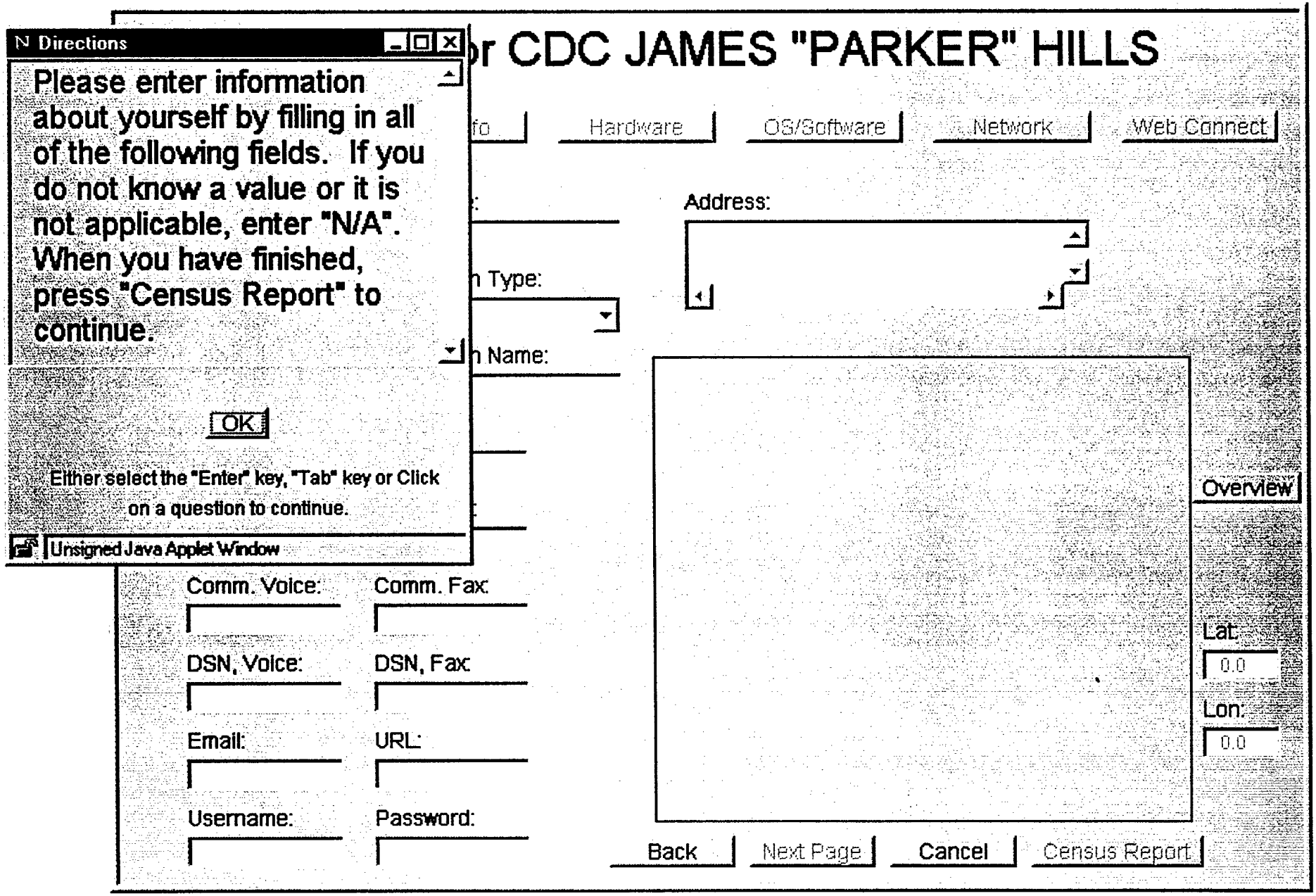
Unsigned Java Applet Window



Lat: 32.0

Lon: -90.0

Back | Next Page | Cancel


Directions
Please enter information about yourself by filling in all of the following fields. If you do not know a value or it is not applicable, enter "N/A". When you have finished, press "Census Report" to continue.
OK
Either select the "Enter" key, "Tab" key or Click on a question to continue.
Unsigned Java Applet Window
r CDC JAMES "PARKER" HILLS
Hardware
OS/Software
Network
Web Connect
Address:
Type:
Name:
Overview
Comm. Voice:
Comm. Fax:
DSN, Voice:
DSN, Fax:
Email:
URL:
Username:
Password:
Lat:
0.0
Lon:
0.0
Back
Next Page
Cancel
Census Report

Success

Information received, thank you for participating.

OK

Login

Please select your type of counter-drug position:

☐ CDC ☑ LEA

User Name: tsmith

Password: **********

OK Cancel

://redwood.nrlssc.navy.mil/DartApplet/DartApplet.html

What's F

ome Search Netscape Print Security Stop

LEA Info for CDC STANLEY TURNER

Info | LEA Info | Hardware | OS/Software | Network | Web Connect

- View Info
- Edit Info
- Add New LEA
- Statistics
- Email LEA

Address: 121 Penn. Ave. Edit

Jurisdiction Name: Washington, DC

State: DC

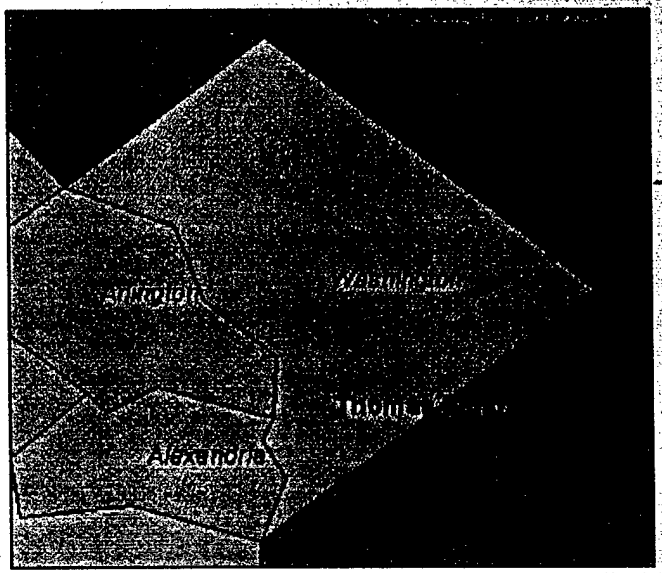
Rank/Title: none

Comm. Fax: 34343

DSN, Fax: 34343

URL: dfdf

Password: washingtondc



Overview

Lat: 38.83

Lon: -77.01

Back Next Page Home Census Report

1. Do you have regular access to a computer to support your LEA duties?

(•) Yes ( ) No

2. Select the type of computer you have access to during your LEA duties.

Common Desktop Personal Computer (PC)

3. What is the functional level or speed of the computer used in your LEA duties?

>= 200Mhz

4. What kind of operating system does your computer use?

Windows NT

5. Is this computer connected to a network?

(•) Yes ( ) No

6. What kind of network connection do you have?

Local Router

7. What speed is your network connection?

56K or greater

8. Does this computer have browser software to connect to the world wide web?

(•) Yes ( ) No

9. What kind of browser software do you have?

MS Internet Explorer

10. Have you ever viewed a webpage using this computer?

(•) Yes ( ) No

Submit

Submit?

Have you checked over all of your information?

Yes, Submit | No, Go Back

LEA Info for CDC STANLEY TURNER

CDC Info | LEA Info | Hardware | OS/Software | Network | Web Connect

LEAs:

- Roy Rogers
- Thomas Smith

LEA Name: Thomas Smith

Jurisdiction Type: State

Jurisdiction Name: Washington, DC

Address: 121 Penn. Ave.

Edit

City: Washington

State: DC

Zip: 20003-1719

Rank/Title: none

Comm. Voice: 3434

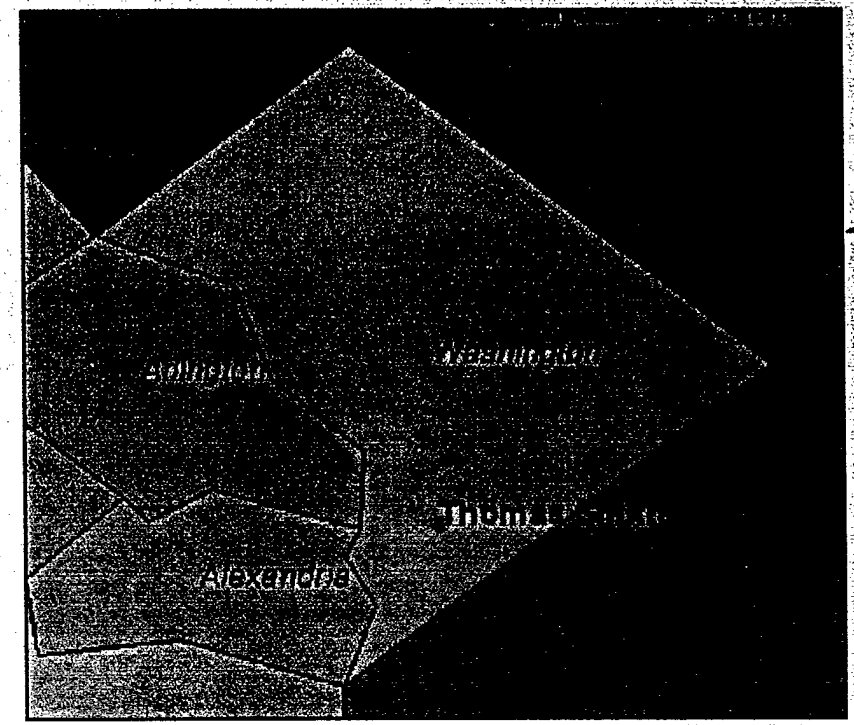
Comm. Fax: 34343

DSN, Voice: 34343

DSN, Fax: 34343

Email: dfdfd

URL: dfdf



Overview

Lat: 38.83!

Lon: -77.01

Back | Next Page | Home | Census Report

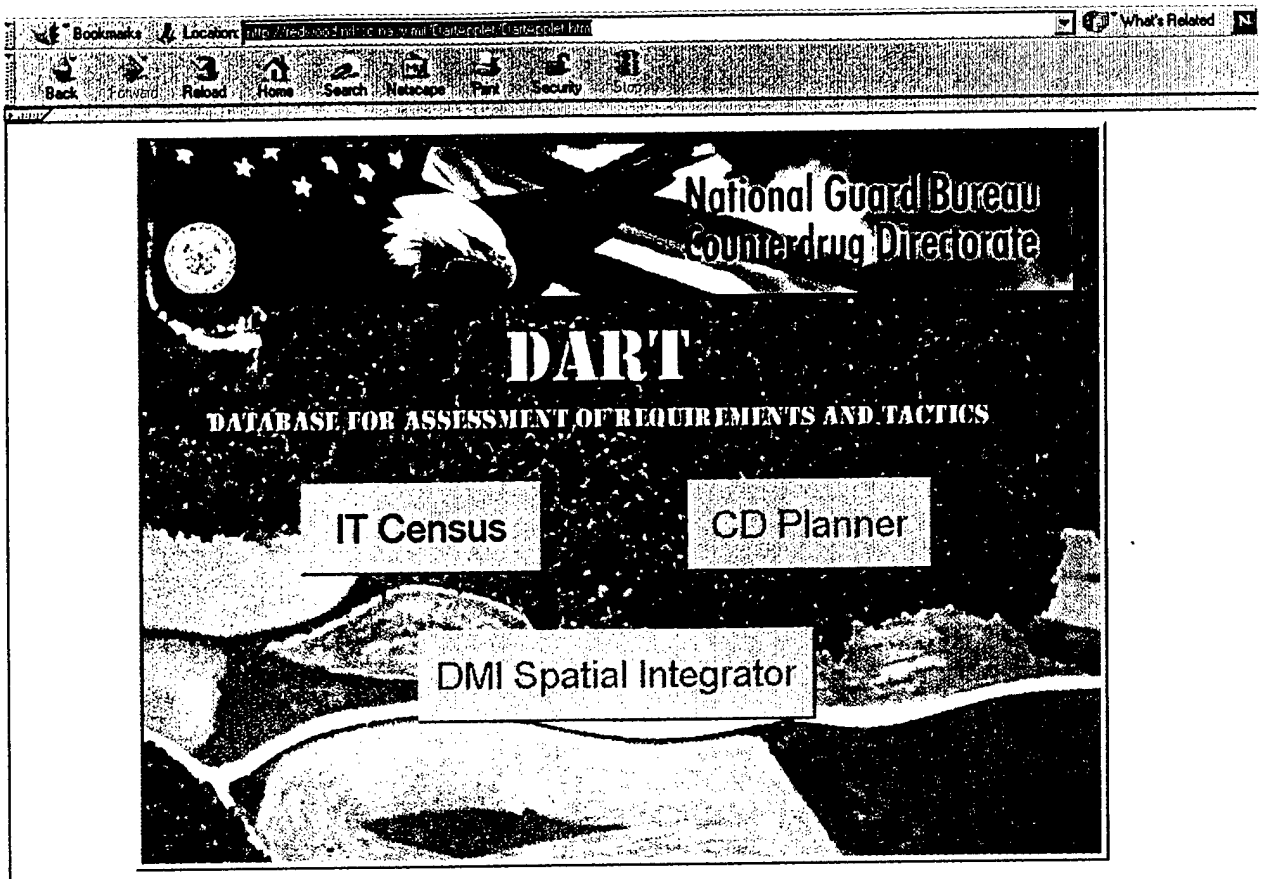
The Database for the Assessment of Requirements and Tactics (DART)

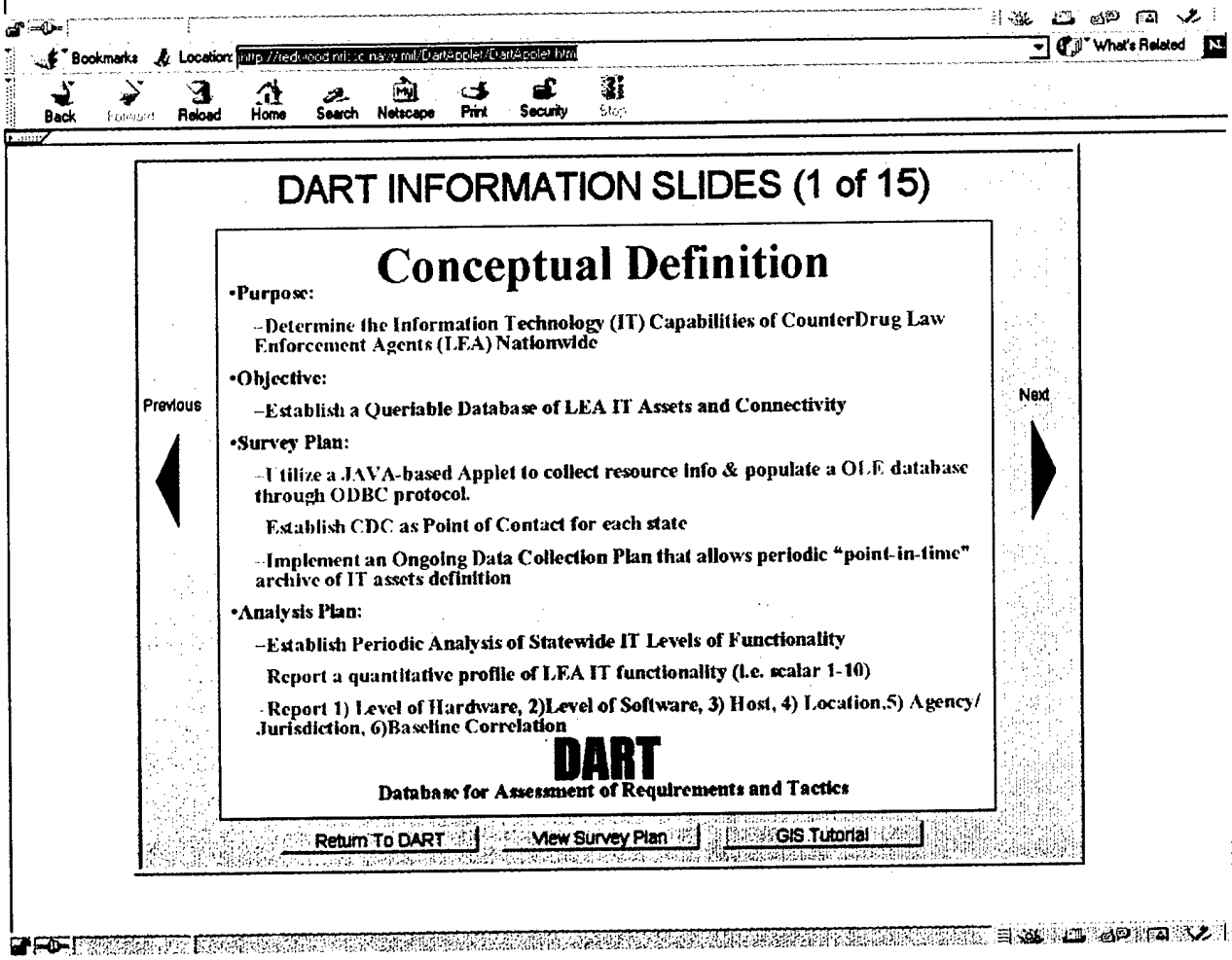
Appendix C: DART WebPages Interface for Program Briefing

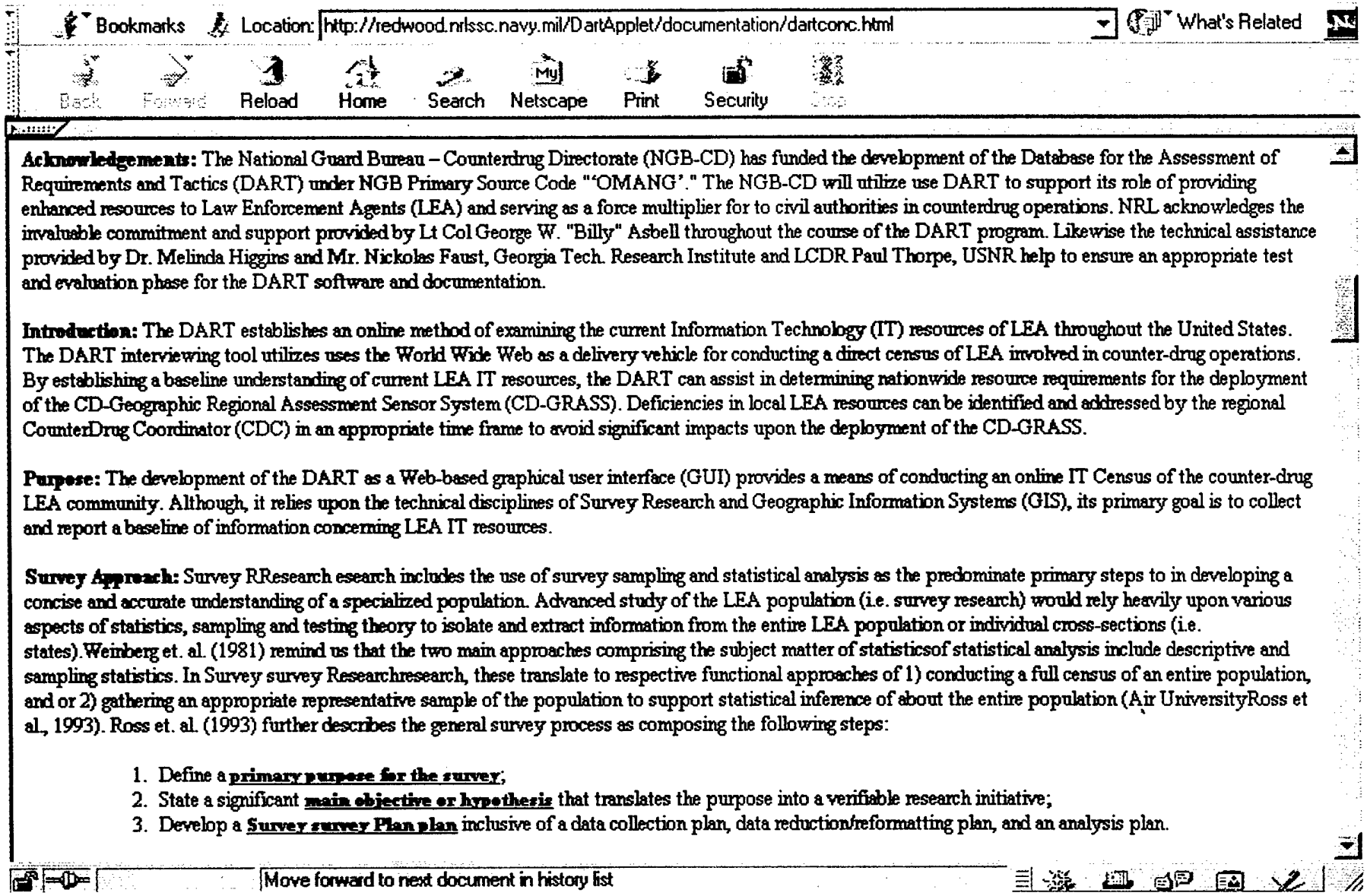
Approved for Public Release; distribution is unlimited.

Appendix C. DART WebPages Interface for Program Briefing

Appendix C offers a graphical presentation of the program briefing screens presented to the DART user. As the DART user accesses the main DART WebPages, located at http://gidb.nrlssc.navy.mil/DartApplet/DartApplet.html, a hyperlink reference to the main DART title block directs the user to briefing information concerning the DART program. The following page cycles through approximately 15 vugraphs to explain the technical approach of the DART program. The following pages offer a printed version of those slides. Also available on this WebPages are links to a GIS Tutorial (re: Appendix D) and the DART Survey Plan (re: an online version of this NRL report).


Bookmarks
Location:
What's Related
Back
Forward
Reload
Home
Search
Netscape
Print
Security
Stop
National Guard Bureau
Counterdrug Directorate
DART
DATABASE FOR ASSESSMENT OF REQUIREMENTS AND TACTICS
IT Census
CD Planner
DMI Spatial Integrator


Bookmarks
Location:
What's Related
Back
Forward
Reload
Home
Search
Netscape
Print
Security
Stop
DART INFORMATION SLIDES (1 of 15)
Conceptual Definition
•Purpose:
–Determine the Information Technology (IT) Capabilities of CounterDrug Law Enforcement Agents (LEA) Nationwide
•Objective:
–Establish a Queriable Database of LEA IT Assets and Connectivity
•Survey Plan:
–Utilize a JAVA-based Applet to collect resource info & populate a OLE database through ODBC protocol.
Establish CDC as Point of Contact for each state
–Implement an Ongoing Data Collection Plan that allows periodic "point-in-time" archive of IT assets definition
•Analysis Plan:
–Establish Periodic Analysis of Statewide IT Levels of Functionality
Report a quantitative profile of LEA IT functionality (i.e. scalar 1-10)
–Report 1) Level of Hardware, 2)Level of Software, 3) Host, 4) Location,5) Agency/ Jurisdiction, 6)Baseline Correlation
DART
Database for Assessment of Requirements and Tactics
Previous
Next
Return To DART
View Survey Plan
GIS Tutorial



Acknowledgements: The National Guard Bureau – Counterdrug Directorate (NGB-CD) has funded the development of the Database for the Assessment of Requirements and Tactics (DART) under NGB Primary Source Code "'OMANG'." The NGB-CD will utilize use DART to support its role of providing enhanced resources to Law Enforcement Agents (LEA) and serving as a force multiplier for to civil authorities in counterdrug operations. NRL acknowledges the invaluable commitment and support provided by Lt Col George W. "Billy" Asbell throughout the course of the DART program. Likewise the technical assistance provided by Dr. Melinda Higgins and Mr. Nickolas Faust, Georgia Tech. Research Institute and LCDR Paul Thorpe, USNR help to ensure an appropriate test and evaluation phase for the DART software and documentation.

Introduction: The DART establishes an online method of examining the current Information Technology (IT) resources of LEA throughout the United States. The DART interviewing tool utilizes uses the World Wide Web as a delivery vehicle for conducting a direct census of LEA involved in counter-drug operations. By establishing a baseline understanding of current LEA IT resources, the DART can assist in determining nationwide resource requirements for the deployment of the CD-Geographic Regional Assessment Sensor System (CD-GRASS). Deficiencies in local LEA resources can be identified and addressed by the regional CounterDrug Coordinator (CDC) in an appropriate time frame to avoid significant impacts upon the deployment of the CD-GRASS.

Purpose: The development of the DART as a Web-based graphical user interface (GUI) provides a means of conducting an online IT Census of the counter-drug LEA community. Although, it relies upon the technical disciplines of Survey Research and Geographic Information Systems (GIS), its primary goal is to collect and report a baseline of information concerning LEA IT resources.

Survey Approach: Survey RResearch esearch includes the use of survey sampling and statistical analysis as the predominate primary steps to in developing a concise and accurate understanding of a specialized population. Advanced study of the LEA population (i.e. survey research) would rely heavily upon various aspects of statistics, sampling and testing theory to isolate and extract information from the entire LEA population or individual cross-sections (i.e. states).Weinberg et. al. (1981) remind us that the two main approaches comprising the subject matter of statisticsof statistical analysis include descriptive and sampling statistics. In Survey survey Researchresearch, these translate to respective functional approaches of 1) conducting a full census of an entire population, and or 2) gathering an appropriate representative sample of the population to support statistical inference of about the entire population (Air UniversityRoss et al., 1993). Ross et. al. (1993) further describes the general survey process as composing the following steps:

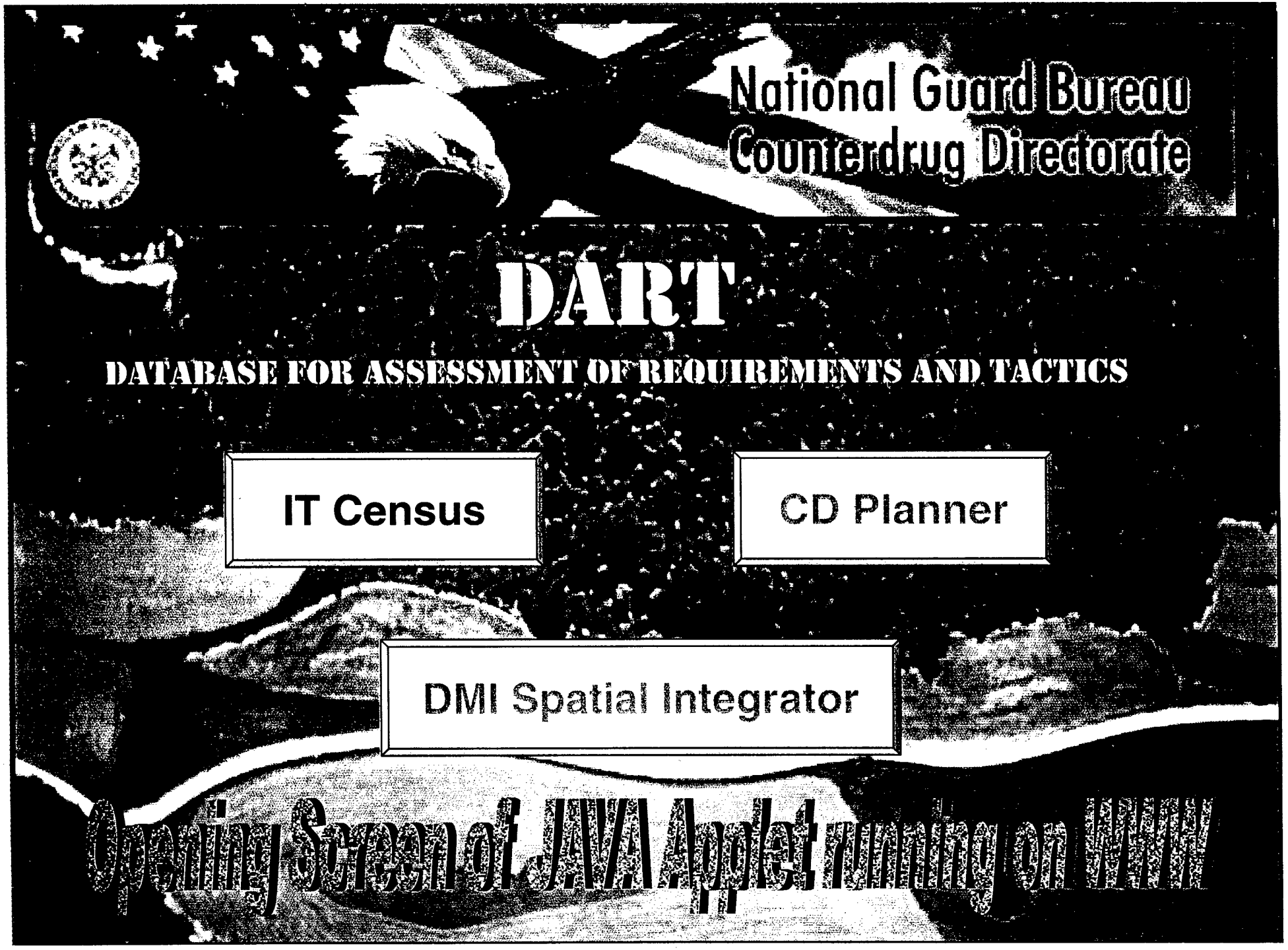
1. Define a **primary purpose for the survey**;
2. State a significant **main objective or hypothesis** that translates the purpose into a verifiable research initiative;
3. Develop a **Survey survey Plan plan** inclusive of a data collection plan, data reduction/reformatting plan, and an analysis plan.

Conceptual Definition

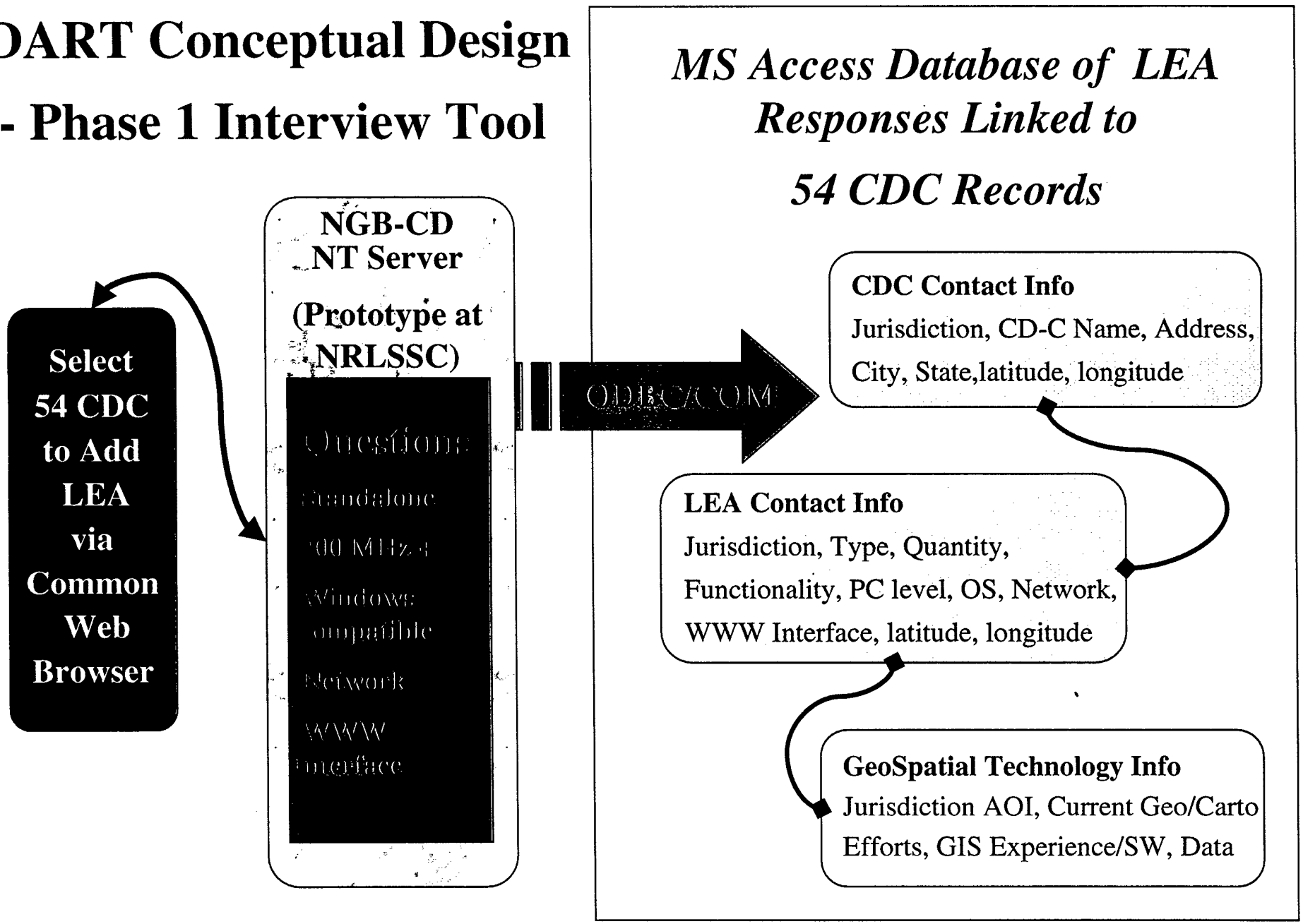
- **Purpose:**
  - –Determine the Information Technology (IT) Capabilities of CounterDrug Law Enforcement Agents (LEA) Nationwide
- **Objective:**
  - –Establish a Queriable Database of LEA IT Assets and Connectivity
- **Survey Plan:**
  - –Utilize a JAVA-based Applet to collect resource info & populate a OLE database through ODBC protocol.
  - –Establish CDC as Point of Contact for each state
  - –Implement an Ongoing Data Collection Plan that allows periodic "point-in-time" archive of IT assets definition
- **Analysis Plan:**
  - –Establish Periodic Analysis of Statewide IT Levels of Functionality
  - –Report a quantitative profile of LEA IT functionality (i.e. scalar 1-10)
  - –Report 1) Level of Hardware, 2)Level of Software, 3) Host, 4) Location,5) Agency/ Jurisdiction, 6)Baseline Correlation

DART

Database for Assessment of Requirements and Tactics


National Guard Bureau
Counterdrug Directorate
DART
DATABASE FOR ASSESSMENT OF REQUIREMENTS AND TACTICS
IT Census
CD Planner
DMI Spatial Integrator
Opening Screen of JAVA Applet running on WWW

DART Conceptual Design - Phase 1 Interview Tool



Finalization of Approach

Direction: ODBC Connectivity for Optimum Data

Applet

- Direct Map Interface
- Direct ODBC Connect
- Map Coverage
- Extensible to other
- Platform for GIS SW
- GCDB Compatibility

ArcView/ Dialog Designer /IMS

- **Direct Map Interface**
- **ODBC Engine**
- **License Requirement**
- **Limited Connect to other GIS Packages**

Application

- Direct Map Interface
- ODBC Connect Engine
- Wide Coverage
- Extensible to other
- IMS Compatibility
- Footprint

Open Front-page Through Common URL/JAVA Applet

Select Mode

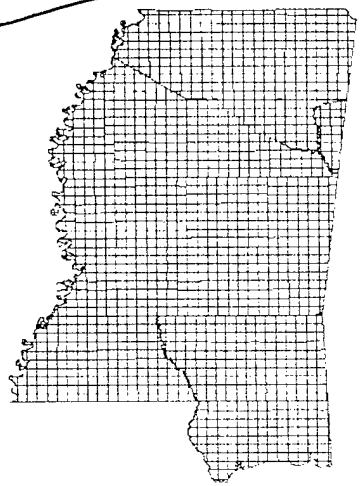
HR Interview Tool
Planning Tool
DMI Tool

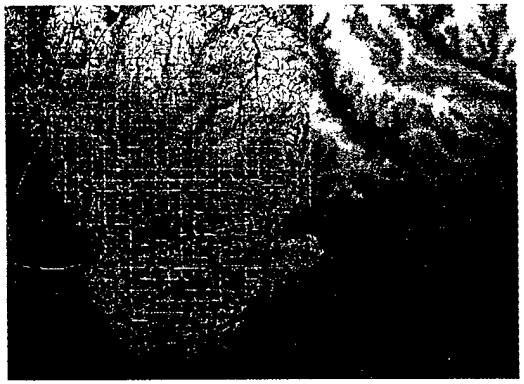
Opening Screen
List and Map
Query/QC CDC
User Info

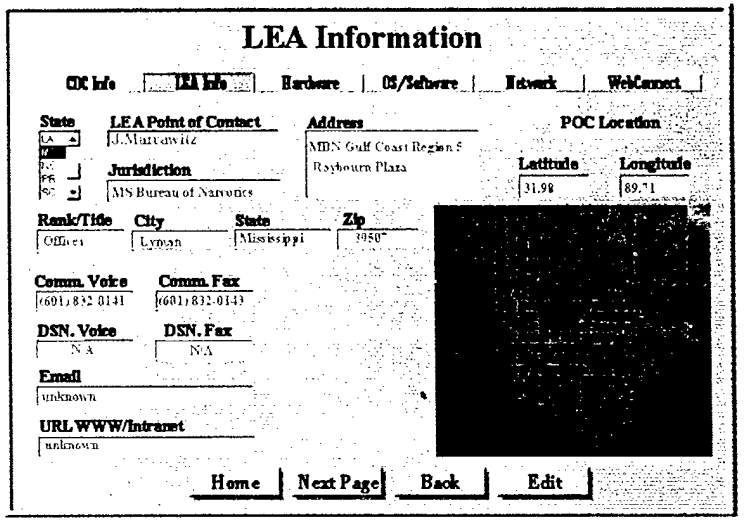
GeoSpatial Tech. Info
LEA Resource Info
CDC Contact Info

Display CDC/LEA Contact Info

and/or

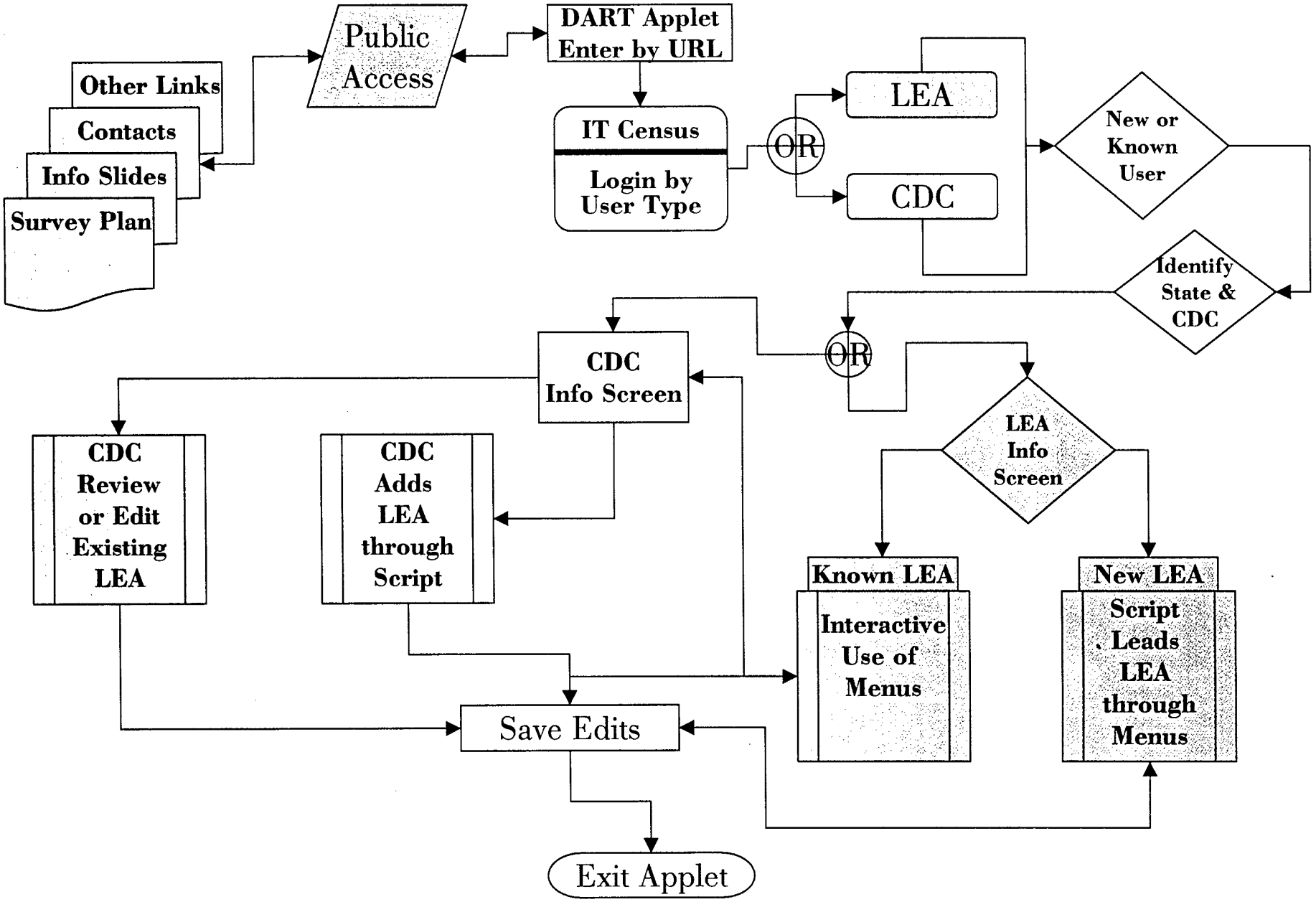






Functional Design - User Interface via JAVA Applet

DART Functional Flow Diagram



1st Level IT Resources Screen

CDC Information

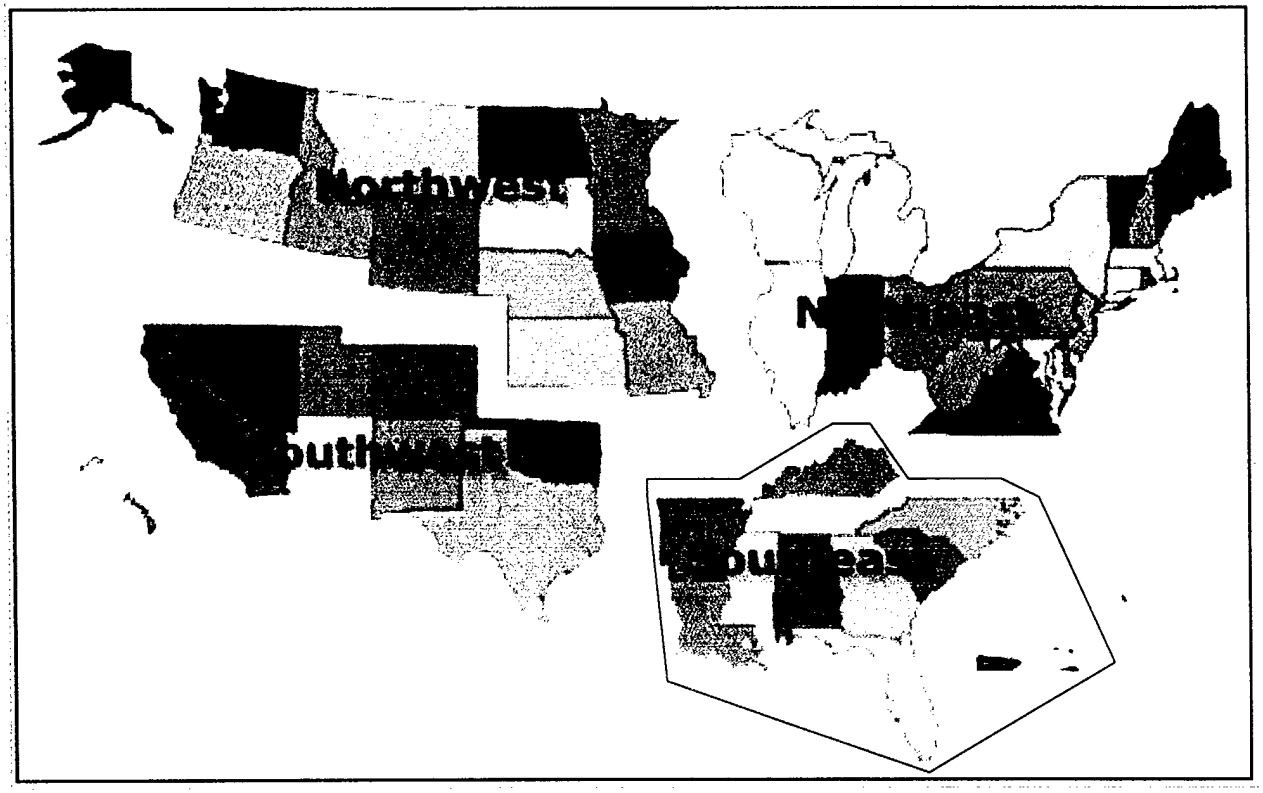
Please select a menu option

CDC Info | LEA Info | Hardware | OS/Software | Network | WebConnect

- **View Info**
- **Edit Info**
- **Email CDC**

LA
MS
NC
PR
SC

JAMES "PARKER" HILLS



Home | Next Page | Back | Edit

1st Level IT Resources Screen

CDC Information

CDC Info | LEA Info | Hardware | OS/Software | Network | WebConnect

State
LA
MS
NC
PR
SC

Current CD Coordinator
JAMES "PARKER" HILLS

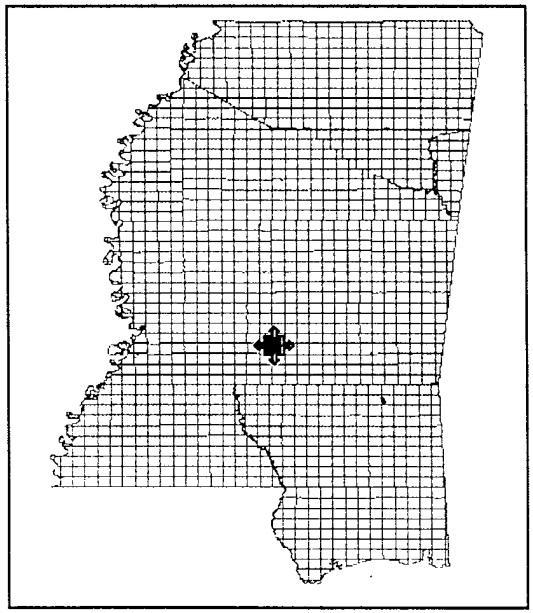
CDC Jurisdiction
Army National Guard

Address
NGMS-OTO-DS,
141 Military Drive

POC Location

Latitude
31.98

Longitude
89.71



Rank/Title
Colonel/COL

City
Jackson

State
Mississippi

Zip
39208-8860

Comm. Voice
(601) 313-1670

Comm. Fax
(601) 313-1673

DSN. Voice
293-1670

DSN. Fax
293-1673

Email
unknown

URLWWW/Intranet
unknown

Home | Next Page | Back | Edit

2nd Level IT Resources Screen

LEA Information

CDC Info | LEA Info | Hardware | OS/Software | Network | WebConnect

State
LA
MS
NC
PR
SC

LEA Point of Contact
J.Marcawitz

Jurisdiction
MS Bureau of Narcotics

Address
MBN Gulf Coast Region 5
Raybourn Plaza

POC Location

Latitude
31.98

Longitude
89.71

Rank/Title
Officer

City
Lyman

State
Mississippi

Zip
39507

Comm. Voice
(601) 832-0141

Comm. Fax
(601) 832-0143

DSN. Voice
N/A

DSN. Fax
N/A

Email
unknown

URLWWW/Intranet
unknown

Home | Next Page | Back | Edit

3rd Level IT Resources Screen

LEA Hardware Information

CDC Info | LEA Info | Hardware | OS/Software | Network | WebConnect

State
LA
MS
NC
PR
SC

LEA Point of Contact
J. Marcawitz

Jurisdiction
MS Bureau of Narcotics

1. Do you have access to a computer during work?

◉ **1. Yes** ○ **2. No**

2. Select the type of computer you have access to at your work place?

Other
IBM PC Compatible
Macintosh
Workstation
Internet PC Terminal

4. What is the power frequency of your computer?

< 200 MHz
= > 200 MHz
Other

Home | Next Page | Back | Edit

4th Level IT Resources Screen

LEA OS/Software Information

CDC Info | LEA Info | Hardware | OS/Software | Network | WebConnect

State

LA
MS
NC
PR
SC

LEA Point of Contact

J. Marcawitz

Jurisdiction

MS Bureau of Narcotics

1. What kind of operating system does your computer use?

- **Windows 95/2000**
- **Windows NT**
- **Macintosh OS**
- **UNIX**
- **Other**

Home | Next Page | Back | Edit

4th Level IT Resources Screen

LEA OS/Software Information

CDC Info | LEA Info | Hardware | OS/Software | Network | WebConnect

State
LA
MS
NC
PR
SC

LEA Point of Contact
J. Marcawitz

Jurisdiction
MS Bureau of Narcotics

1. Is this computer connected to a network?

◉ 1. Yes ○ 2. No

2. What kind of network connection do you have?

- Modem over Phone Line
- Local Router
- High Speed Multiplexer
- Other
- Unknown

3. What speed is your network connection?

- 56K or greater
- T1 or greater
- Other
- Unknown

Home | Next Page | Back | Edit

5th Level IT Resources Screen

LEA OS/Software Information

CDC Info | LEA Info | Hardware | OS/Software | Network | WebConnect

State
LA
MS
NC
PR
SC

LEA Point of Contact
J. Marcawitz

Jurisdiction
MS Bureau of Narcotics

1. Does this computer have browser software to connect to the World Wide Web?

◉ **1. Yes** ○ **2. No**

2. What kind of browser software do you have?

Netscape Navigator
MS Internet Explorer
Other
Unknown

1. Have you ever viewed a webpage using this computer?

◉ **1. Yes** ○ **2. No**

Home | Next Page | Back | Edit

Functional System Requirements

Data Collection	Management & Retrieval	Analysis	Output	End-user Application
•54 Individual CDC login web URL •Go right to state •Answer point & click static questions •Provide user comment fields	•Isolate Based on State •Single Master or 54 separate?	•Select Buttons to Report Critical Limits •Review Matrix or map showing assets and connection to surrounding LEA	•Request IT Resource Report •Select Interactive Maps/Graphs •Go/Stop Connect Matrix •Export Resource Table	•CDC use in Interactive Search/Browser •Import Resource Table into MS Office

Operational System Requirements tied to SW/HW/People

Data Collection	Management & Retrieval	Analysis	Output	End-User Application
•Internet/Common Browser (NetScape/ MS Explorer) •Select Boxes w Dialog Boxes for Comments •Standard 3 button Mouse Functions •ODBC Import (.dbf, xls, txt)	•JAVA Applet •Common SQL code	•Viewshed type •Distance/Zonal	•Default Printer •Table Export using •Access/Excel or ASCII dump	•NT Win based

The Database for the Assessment of Requirements and Tactics (DART)

Appendix D: DART WebPages Interface for GIS Tutorial

Approved for Public Release; distribution is unlimited.

Appendix D. DART WebPages Interface for GIS-Tutorial

Appendix D offers a graphical presentation of the GIS Tutorial. It represents a prototype demonstration of what might serve as an online resource for the NGB-CD's Digital Mapping Initiative (DMI). The tutorial would be expanding resource that DMI could refer LEA and other individuals to as an education reference for developing an understanding of the basic principals of GIS.

GIS Tutorial Slides (1 of 2)

What is GIS?

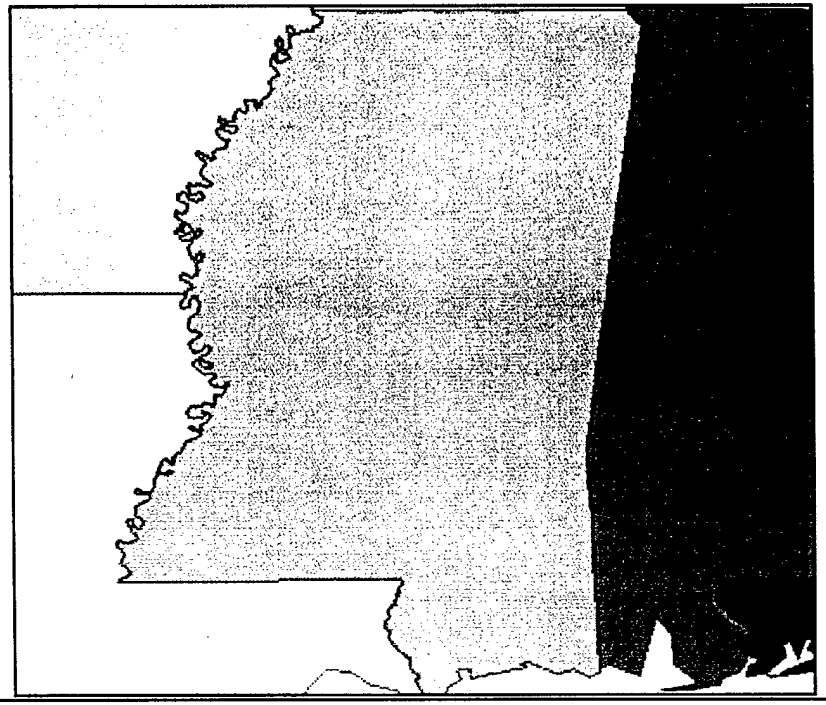
A computer-based approach to managing data by maintaining a reference to its geographic location.

Previous

Next

This spatial data is used by the GIS to produce information supportive of a specialized mission like CounterDrug Operations.

Spatial data are usually represented in a GIS as thematic layers of information that can be overlaid to examine their relationships .



Click on the buttons to turn layers on and off.

- State Boundary
- County Boundaries
- Roads
- Rivers

Return To DART

GIS Tutorial Slides (1 of 2)

What is GIS?

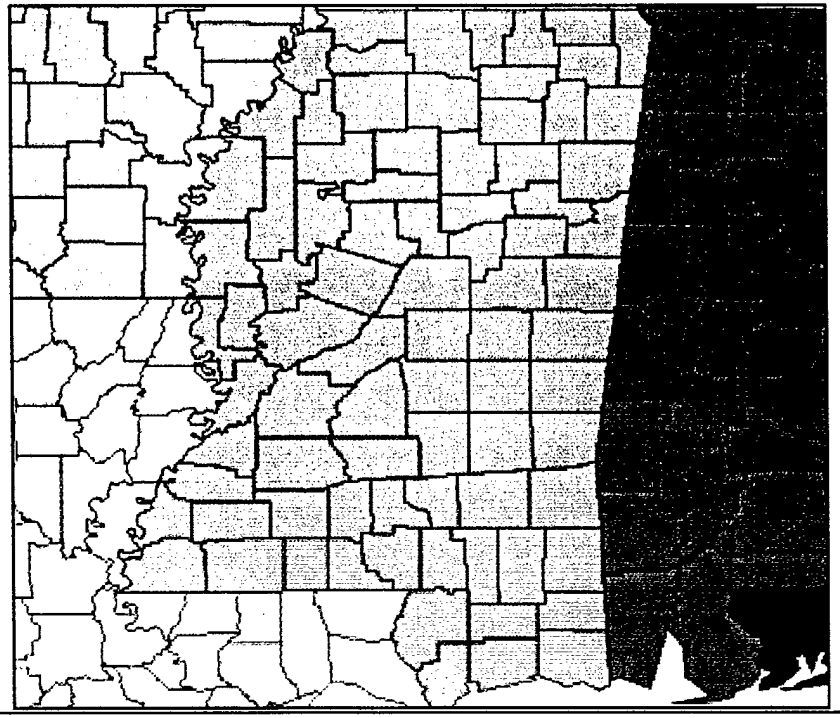
A computer-based approach to managing data by maintaining a reference to its geographic location.

Previous

Next

This spatial data is used by the GIS to produce information supportive of a specialized mission like CounterDrug Operations.

Spatial data are usually represented in a GIS as thematic layers of information that can be overlaid to examine their relationships .



Click on the buttons to turn layers on and off.

- State Boundary
- County Boundaries
- Roads
- Rivers

Return To DART

GIS Tutorial Slides (1 of 2)

What is GIS?

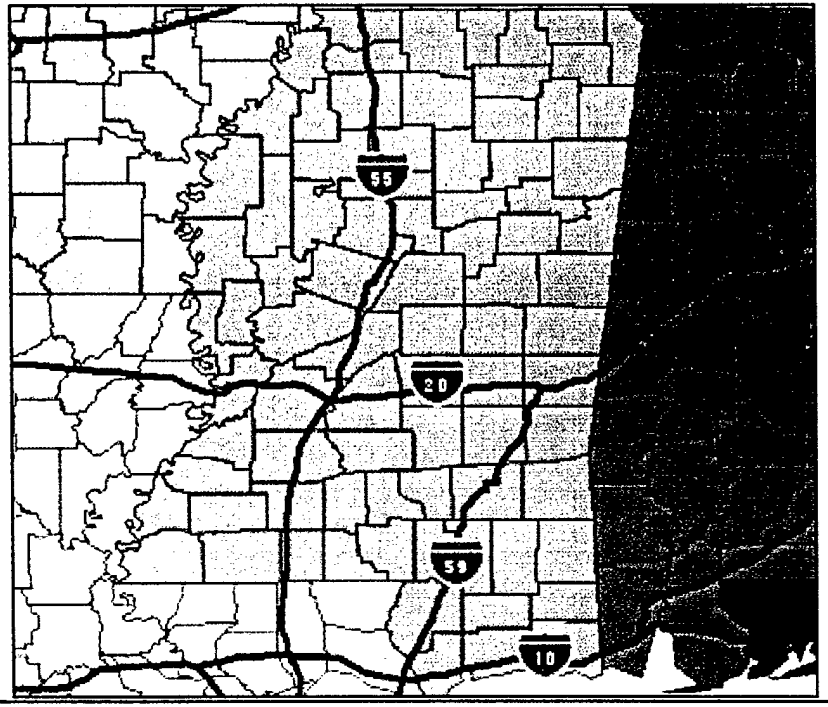
A computer-based approach to managing data by maintaining a reference to its geographic location.

Previous

Next

This spatial data is used by the GIS to produce information supportive of a specialized mission like CounterDrug Operations.

Spatial data are usually represented in a GIS as thematic layers of information that can be overlaid to examine their relationships .



Click on the buttons to turn layers on and off.

- State Boundary
- County Boundaries
- Roads
- Rivers

Return To DART

GIS Tutorial Slides (1 of 2)

What is GIS?

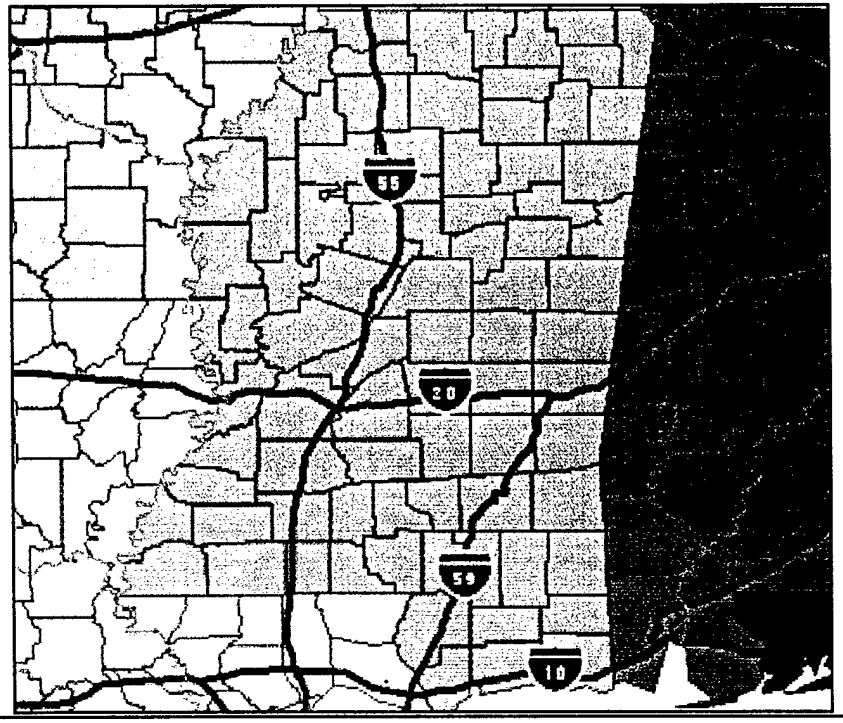
A computer-based approach to managing data by maintaining a reference to its geographic location.

Previous

Next

This spatial data is used by the GIS to produce information supportive of a specialized mission like CounterDrug Operations.

Spatial data are usually represented in a GIS as thematic layers of information that can be overlaid to examine their relationships .



Click on the buttons to turn layers on and off.

- State Boundary
- County Boundaries
- Roads
- Rivers

Return To DART

GIS Tutorial Slides (2 of 2)

Previous

Next

WWW Resources Describing GIS

Click on the buttons to display webpages.

- What is GIS?
- What kinds of data are available?
- Who are some of the major commercial GIS providers?
- What's the future of GIS? Linking Commercial and Public GIS solutions together.

Return To DART


GIS.com--Your Internet Guide to GIS (Geographic Information Systems) - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://www.gis.com/
What's Related
Back Forward Reload Home Search Netscape Print Security Stop
GIS.com
your Internet guide to
Geographic Information Systems
About GIS.com
GIS Day
Geography Network
GIS Jump Station
WHAT IS GIS?
GIS FOR YOUR SPECIALTY
TRY GIS FOR YOURSELF
GIS SOFTWARE
DATA FOR YOUR GIS
EDUCATION & TRAINING
ADDITIONAL RESOURCES
NEWS/EVENTS/TRENDS
ONLINE STORE
Welcome to GIS.com
...your source for information and advice on all aspects of geographic information systems (GIS). Learn what GIS is and can do for you, try it for yourself, find resources for building a GIS, and get the latest news affecting the GIS industry.
Document: Done

GIS Data Depot - Free GIS Data - Netscape

File Edit View Go Communicator Help

Bookmarks Netsite: http://www.gisdatadepot.com/ What's Related

Back Forward Reload Home Search Netscape Print Security Stop

geo COMMUNITY

The GI Mall Store

Your one-stop source for Geographic, Earth, Mapping Information

HOME | SITEMAP | CONTACT | FAQ | ADVERTISE

SOFTWARE DATA NEWS FORUM FUN

SERVING • ABOUT • ACCESS • NEW • HELP • REQUESTS

GIS DATA DEPOT

GeoSearch GO

GeoCommunity

Other GeoSites

SpatialNews Subscription
Enter your email: GO

Geo-Industry Links

READ A BOOK

The GIS Data Depot™ - the largest FREE geo-spatial data repository on the web!

30% Off
USGS DRG Bundles

NOW 30% OFF!

Wednesday, July 26, 2000

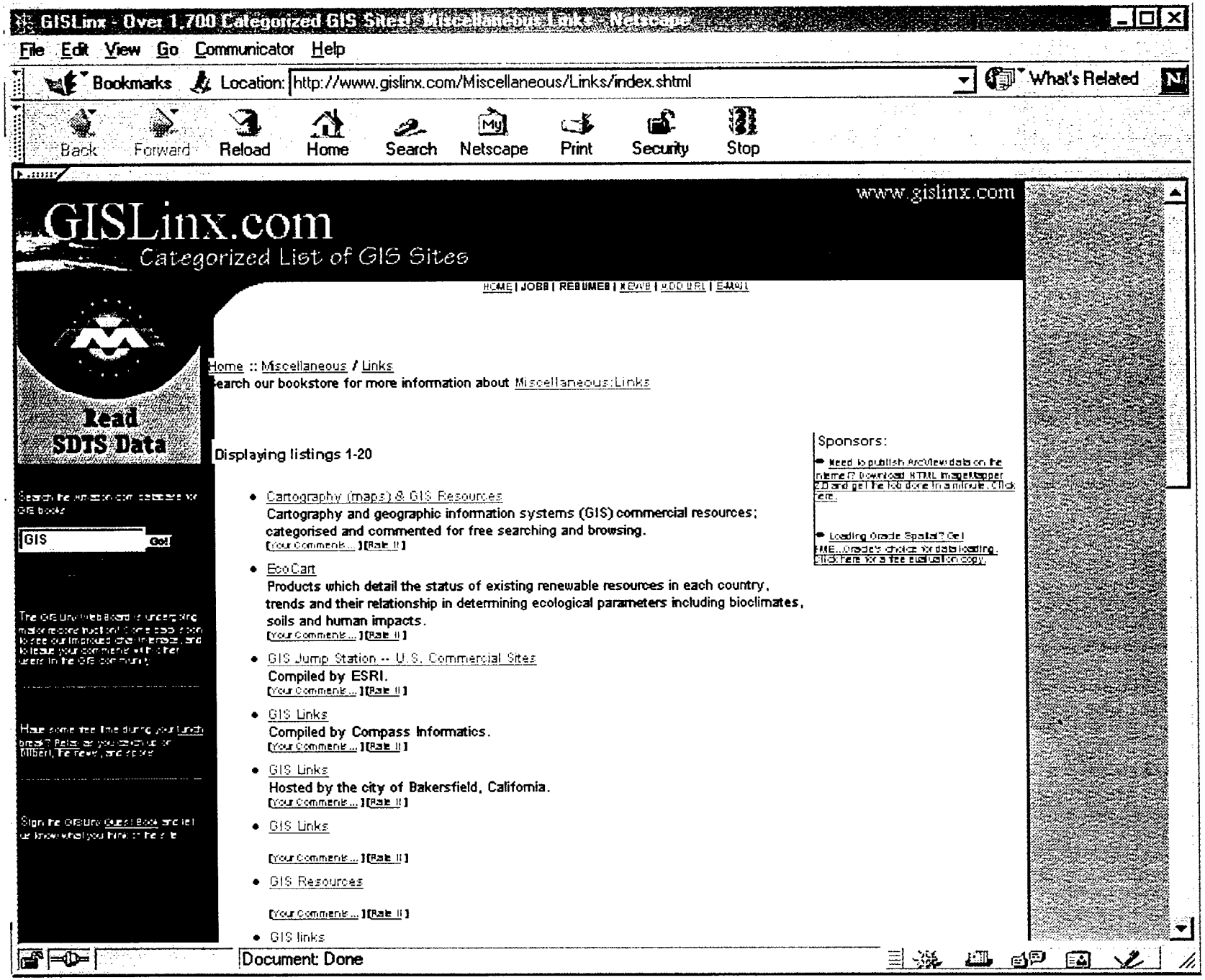
ON SALE Surplus DRGs!
DCW SALE 49.95!

Welcome to The GeoCommunity's™ GIS Data Depot™. This is the location to access our FREE downloads and custom CD creation. Help yourself to data from our online inventory of more than a terabyte of geospatial data including Over 46,000 24K, 25K, 63K, 100k AND 250k USGS DRGs.

FREE DATA ENTER HERE

You'll also find tons of DOQQ, NWI, DTED, and DCW data!

Document: Done


GISLinx - Over 1,700 Categorized GIS Sites - Miscellaneous Links - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://www.gislinx.com/Miscellaneous/Links/index.shtml
What's Related
Back Forward Reload Home Search Netscape Print Security Stop
www.gislinx.com
GISLinx.com
Categorized List of GIS Sites
HOME | JOBS | RESUMES | ADD URL | EMAIL
Read SDTS Data
Home :: Miscellaneous / Links
Search our bookstore for more information about Miscellaneous:Links
Displaying listings 1-20
Sponsors:
Cartography (maps) & GIS Resources
Cartography and geographic information systems (GIS) commercial resources; categorised and commented for free searching and browsing.
[Your Comments...] [Rate It]
EcoCart
Products which detail the status of existing renewable resources in each country, trends and their relationship in determining ecological parameters including bioclimates, soils and human impacts.
[Your Comments...] [Rate It]
GIS Jump Station -- U.S. Commercial Sites
Compiled by ESRI.
[Your Comments...] [Rate It]
GIS Links
Compiled by Compass Informatics.
[Your Comments...] [Rate It]
GIS Links
Hosted by the city of Bakersfield, California.
[Your Comments...] [Rate It]
GIS Links
[Your Comments...] [Rate It]
GIS Resources
[Your Comments...] [Rate It]
GIS links
Document: Done

DMAP Home Page - Netscape

File Edit View Go Communicator Help

Bookmarks Location: http://postoffice.nrlssc.navy.mil/dmap/home.html What's Related

Back Forward Reload Home Search Netscape Print Security Stop

DMAP

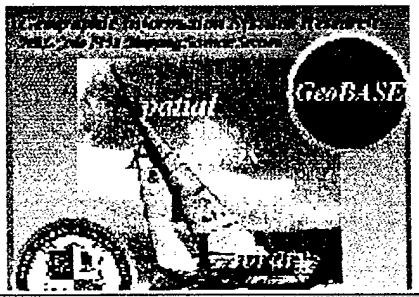
Digital Mapping, Charting & Geodesy Analysis Program

DMAP

About DMAP
Naval Research Laboratory
Code 7440.2
Stennis Space Center, MS 39529

Geospatial Information Database (GIDB)
The Geospatial Information Database (GIDB), developed by the Naval Research Laboratory for the US Marine Corps Warfighting Lab's Urban Warrior Exercise, is an integrated object-oriented digital mapping database.

3D Synthetic Environment Research (VPF+)
DMAP, in conjunction with the University of New Orleans (UNO), has been investigating the development of a three-dimensional spatial data model useful for modeling and simulation applications. As a result of this research, we have developed a new data structure, VPF+.



Spatial Analysis & Integration Library (SAIL)
Mapping GIS Solutions to Real-World Naval Problems

Document: Done

The Database for the Assessment of Requirements and Tactics (DART)

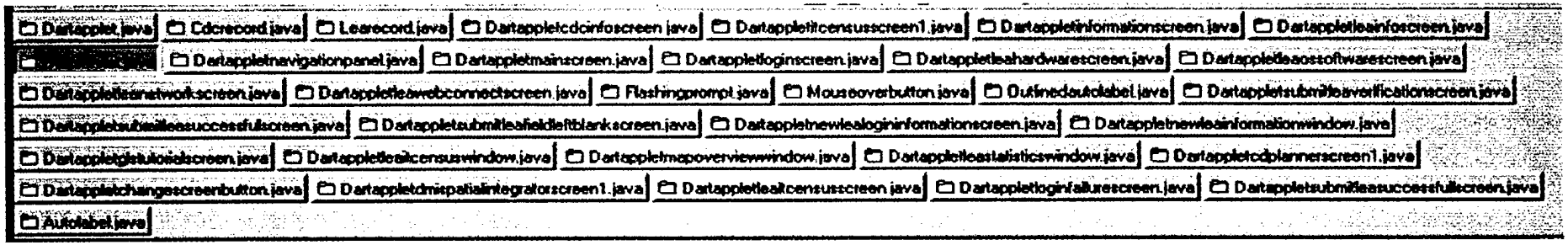
Appendix E: JAVA™ and Avenue™ Source Code

Approved for Public Release; distribution is unlimited.

Appendix E. DART JAVA/Avenue™ Source Code

Appendix E offers hardcopy print out of the source code developed for the DART program. It provides both JAVA™ Version 1.1.8 and the ArcView® Version 3.2 Avenue™ code. The following graphic lists the file names for all the JAVA code, which was written by Mr. Frank McCreedy, NRL Code 7440.2. The remaining pages provide the Avenue™ code was written by Mr. Ralph Perniciaro, Planning Systems, Inc. All DART source code is considered public domain.

Listing of all JAVA code file names as printed in remaining pages of Appendix E.



```
java.applet.*;
java.awt.*;
ava.net.*;
ava.io.*;
java.util.*;

lass DartApplet extends Applet{
largeButtonFont;
screenTitleFont;
smallerScreenTitleFont;
smallestScreenTitleFont;
regularScreenFont;

AppletNavigationPanel navigationPanel;

AppletMainScreen scrMainScreen;
AppletITCensusScreen1 scrITCensusScreen1;
AppletInformationScreen scrDartInformationScreen;
AppletGISTutorialScreen scrDartGISTutorialScreen;
AppletCDPlannerScreen1 scrCDPlannerScreen1;
AppletDMISpatialIntegratorScreen1 scrDMISpatialIntegratorScreen1;
AppletLoginScreen scrLoginScreen;
AppletLoginFailureScreen scrLoginFailureScreen;
AppletNewLEALoginInformationScreen scrNewLEALoginInformationScreen;
AppletCDCInfoScreen scrCDCInfoScreen;
AppletLEAInfoScreen scrLEAInfoScreen;
AppletLEAHardwareScreen scrLEAHardwareScreen;
AppletLEAOSSoftwareScreen scrLEAOSSoftwareScreen;
AppletLEANetworkScreen scrLEANetworkScreen;
AppletLEAWebConnectScreen scrLEAWebConnectScreen;
tAppletSubmitLEAVerificationScreen scrSubmitLEAVerificationScreen;
tAppletSubmitLEASuccessfulScreen scrSubmitLEASuccessfulScreen;
AppletSubmitLEAFieldLeftBlankScreen scrSubmitLEAFieldLeftBlankScreen;
AppletNewLEAInformationWindow wndNewLEAInformationWindow;
tAppletMapOverviewWindow wndMapOverviewWindow;
AppletLEAITCensusWindow wndLEAITCensusWindow;
AppletLEAStatisticsWindow wndLEAStatisticsWindow;

lean blnInNewLEAMode = false;
ean blnNewLEAHasGoneThroughScript = false;
ng strCurrentLoggedInUser = null;
ing strCurrentLoggedInUserPassword = null;
ing strCurrentLoggedInUserState = null;
ng strCurrentLoggedInUserCDCUsername = null;
ng strCurrentLoggedInUserCDCName = null;
Record lrNewLEARecord = null;

table htblStatesOfSouthwestRegion;
table htblStatesOfSoutheastRegion;
htable htblStatesOfNorthwestRegion;
table htblStatesOfNortheastRegion;

ShingPrompt fpNextItemPrompt;

ic void showScreen(Panel screenToShow){
ndNewLEAInformationWindow.setVisible(false);
wndLEAITCensusWindow.setVisible(false);
wndMapOverviewWindow.setVisible(false);
ndLEAStatisticsWindow.setVisible(false);
crMainScreen.setVisible(false);
scrITCensusScreen1.setVisible(false);
crDartInformationScreen.setVisible(false);
crDartGISTutorialScreen.setVisible(false);
crCDPlannerScreen1.setVisible(false);
scrDMISpatialIntegratorScreen1.setVisible(false);
```

```
LoginScreen.setVisible(false);
LoginFailureScreen.setVisible(false);
CDCInfoScreen.setVisible(false);
LEAInfoScreen.setVisible(false);
LEAHardwareScreen.setVisible(false);
LEAOSSoftwareScreen.setVisible(false);
LEANetworkScreen.setVisible(false);
LEAWebConnectScreen.setVisible(false);
SubmitLEAVerificationScreen.setVisible(false);
SubmitLEASuccessfulScreen.setVisible(false);
SubmitLEAFieldLeftBlankScreen.setVisible(false);

( (screenToShow == scrDartInformationScreen) || (screenToShow == scrDartGISTutorial
|
  (screenToShow == scrLoginScreen) || (screenToShow == scrLoginFailureScreen) ||
  (screenToShow == scrSubmitLEAVerificationScreen) || (screenToShow == scrSubmitLEA
lScreen) ||
  (screenToShow == scrSubmitLEAFieldLeftBlankScreen) ){
navigationPanel.setVisible(false);

e{
navigationPanel.setVisible(true);

eenToShow.setVisible(true);


 void init(){
Layout(null);
Background(Color.white);

geButtonFont = new Font("Symbol", Font.PLAIN, 30);
eenTitleFont = new Font("Symbol", Font.PLAIN, 30);
llerScreenTitleFont = new Font("Symbol", Font.PLAIN, 20);
llestScreenTitleFont = new Font("Symbol", Font.BOLD, 14);
ularScreenFont = new Font("Symbol", Font.PLAIN, 14);

extItemPrompt = new FlashingPrompt();
extItemPrompt.setSize(20, 20);
extItemPrompt.setVisible(true);
extItemPrompt.startFlashing();

igationPanel = new DartAppletNavigationPanel(this);
igationPanel.setLocation(50, 70);
igationPanel.setVisible(false);
(navigationPanel);

MainScreen = new DartAppletMainScreen(this);
MainScreen.setSize(getSize().width, getSize().height);
MainScreen.setLocation(0, 0);
MainScreen.setVisible(false);
(scrMainScreen);

ITCensusScreen1 = new DartAppletITCensusScreen1(this);
ITCensusScreen1.setSize(getSize().width, getSize().height);
ITCensusScreen1.setLocation(0, 0);
ITCensusScreen1.setVisible(false);
(scrITCensusScreen1);

CDCInfoScreen = new DartAppletCDCInfoScreen(this);
CDCInfoScreen.setSize(getSize().width, getSize().height);
CDCInfoScreen.setLocation(0, 0);
CDCInfoScreen.setVisible(false);
(scrCDCInfoScreen);

LEAInfoScreen = new DartAppletLEAInfoScreen(this);
```

```
        scrLEAInfoScreen.setSize(getSize().width, getSize().height);
        scrLEAInfoScreen.setLocation(0, 0);
        scrLEAInfoScreen.setVisible(false);
        add(scrLEAInfoScreen);

        scrLEAHardwareScreen = new DartAppletLEAHardwareScreen(this);
        scrLEAHardwareScreen.setSize(getSize().width, getSize().height);
        scrLEAHardwareScreen.setLocation(0, 0);
        scrLEAHardwareScreen.setVisible(false);
        add(scrLEAHardwareScreen);

        scrLEAOSSoftwareScreen = new DartAppletLEAOSSoftwareScreen(this);
        scrLEAOSSoftwareScreen.setSize(getSize().width, getSize().height);
        scrLEAOSSoftwareScreen.setLocation(0, 0);
        scrLEAOSSoftwareScreen.setVisible(false);
        add(scrLEAOSSoftwareScreen);

        scrLEANetworkScreen = new DartAppletLEANetworkScreen(this);
        scrLEANetworkScreen.setSize(getSize().width, getSize().height);
        scrLEANetworkScreen.setLocation(0, 0);
        scrLEANetworkScreen.setVisible(false);
        add(scrLEANetworkScreen);

        scrLEAWebConnectScreen = new DartAppletLEAWebConnectScreen(this);
        scrLEAWebConnectScreen.setSize(getSize().width, getSize().height);
        scrLEAWebConnectScreen.setLocation(0, 0);
        scrLEAWebConnectScreen.setVisible(false);
        add(scrLEAWebConnectScreen);

        scrDartInformationScreen = new DartAppletInformationScreen(this);
        scrDartInformationScreen.setSize(getSize().width, getSize().height);
        scrDartInformationScreen.setLocation(0, 0);
        scrDartInformationScreen.setVisible(false);
        add(scrDartInformationScreen);

        scrDartGISTutorialScreen = new DartAppletGISTutorialScreen(this);
        scrDartGISTutorialScreen.setSize(getSize().width, getSize().height);
        scrDartGISTutorialScreen.setLocation(0, 0);
        scrDartGISTutorialScreen.setVisible(false);
        add(scrDartGISTutorialScreen);

        scrCDPlannerScreen1 = new DartAppletCDPlannerScreen1(this);
        scrCDPlannerScreen1.setSize(getSize().width, getSize().height);
        scrCDPlannerScreen1.setLocation(0, 0);
        scrCDPlannerScreen1.setVisible(false);
        add(scrCDPlannerScreen1);

        scrDMISpatialIntegratorScreen1 = new DartAppletDMISpatialIntegratorScreen1(this);
        scrDMISpatialIntegratorScreen1.setSize(getSize().width, getSize().height);
        scrDMISpatialIntegratorScreen1.setLocation(0, 0);
        scrDMISpatialIntegratorScreen1.setVisible(false);
        add(scrDMISpatialIntegratorScreen1);

        scrLoginScreen = new DartAppletLoginScreen(this);
        scrLoginScreen.setSize((int)(getSize().width/3.0 + 0.5), (int)(getSize().height/1.7 +
0.5));
        scrLoginScreen.setLocation( (int)(getSize().width/2.0 - scrLoginScreen.getSize().width
/2.0 + 0.5),
                                    (int)(getSize().height/2.0 - scrLoginScreen.getSize().heig
ht/2.0 + 0.5));
        scrLoginScreen.setVisible(false);
        add(scrLoginScreen);

        scrLoginFailureScreen = new DartAppletLoginFailureScreen(this);
        scrLoginFailureScreen.setSize((int)(getSize().width/3.0 + 0.5), (int)(getSize().height
```

```
/2.0 + 0.5));
      scrLoginFailureScreen.setLocation( (int)(getSize().width/2.0 - scrLoginFailureScreen.g
etSize().width/2.0 + 0.5),
                                         (int)(getSize().height/2.0 - scrLoginFailureScreen.
getSize().height/2.0 + 0.5));
      scrLoginFailureScreen.setVisible(false);
      add(scrLoginFailureScreen);

      scrNewLEALoginInformationScreen = new DartAppletNewLEALoginInformationScreen(this);
      scrNewLEALoginInformationScreen.setSize((int)(getSize().width/2.6 + 0.5), (int)(getSiz
e().height/1.9 + 0.5));
      scrNewLEALoginInformationScreen.setLocation( (int)(getSize().width/2.0 - scrNewLEALogi
nInformationScreen.getSize().width/2.0 + 0.5),
                                         (int)(getSize().height/2.0 - scrNewLEALoginInformat
ionScreen.getSize().height/2.0 + 0.5));
      scrNewLEALoginInformationScreen.setVisible(false);
      add(scrNewLEALoginInformationScreen);

      scrSubmitLEAVerificationScreen = new DartAppletSubmitLEAVerificationScreen(this);
      scrSubmitLEAVerificationScreen.setSize((int)(getSize().width/3.0 + 0.5), (int)(getSize
().height/2.0 + 0.5));
      scrSubmitLEAVerificationScreen.setLocation( (int)(getSize().width/2.0 - scrSubmitLEAVe
rificationScreen.getSize().width/2.0 + 0.5),
                                                  (int)(getSize().height/2.0 - scrSubmitLEAV
erificationScreen.getSize().height/2.0 + 0.5));
      scrSubmitLEAVerificationScreen.setVisible(false);
      add(scrSubmitLEAVerificationScreen);

      scrSubmitLEASuccessfulScreen = new DartAppletSubmitLEASuccessfulScreen(this);
      scrSubmitLEASuccessfulScreen.setVisible(false);
      scrSubmitLEASuccessfulScreen.setSize((int)(getSize().width/3.0 + 0.5), (int)(getSize()
.height/2.0 + 0.5));
      scrSubmitLEASuccessfulScreen.setLocation( (int)(getSize().width/2.0 - scrSubmitLEASucc
essfulScreen.getSize().width/2.0 + 0.5),
                                                  (int)(getSize().height/2.0 - scrSubmitLEAS
uccessfulScreen.getSize().height/2.0 + 0.5));
      scrSubmitLEASuccessfulScreen.setVisible(false);
      add(scrSubmitLEASuccessfulScreen);

      scrSubmitLEAFieldLeftBlankScreen = new DartAppletSubmitLEAFieldLeftBlankScreen(this);
      scrSubmitLEAFieldLeftBlankScreen.setVisible(false);
      scrSubmitLEAFieldLeftBlankScreen.setSize((int)(getSize().width/3.0 + 0.5), (int)(getSi
ze().height/2.0 + 0.5));
      scrSubmitLEAFieldLeftBlankScreen.setLocation( (int)(getSize().width/2.0 - scrSubmitLEA
FieldLeftBlankScreen.getSize().width/2.0 + 0.5),
                                                  (int)(getSize().height/2.0 - scrSubmitLEAF
ieldLeftBlankScreen.getSize().height/2.0 + 0.5));
      scrSubmitLEAFieldLeftBlankScreen.setVisible(false);
      add(scrSubmitLEAFieldLeftBlankScreen);

      wndNewLEAInformationWindow = new DartAppletNewLEAInformationWindow(this);
      wndNewLEAInformationWindow.setVisible(false);

      wndLEAITCensusWindow = new DartAppletLEAITCensusWindow(this);
      wndLEAITCensusWindow.setVisible(false);

      wndMapOverviewWindow = new DartAppletMapOverviewWindow(this);
      wndMapOverviewWindow.setVisible(false);

      wndLEAStatisticsWindow = new DartAppletLEAStatisticsWindow(this);
      wndLEAStatisticsWindow.setVisible(false);

      htblStatesOfSoutheastRegion = new Hashtable();
      htblStatesOfSoutheastRegion.put("AL", " ");
      htblStatesOfSoutheastRegion.put("AR", " ");
```

```
        htblStatesOfSoutheastRegion.put("FL", " ");
        htblStatesOfSoutheastRegion.put("GA", " ");
        htblStatesOfSoutheastRegion.put("KY", " ");
        htblStatesOfSoutheastRegion.put("LA", " ");
        htblStatesOfSoutheastRegion.put("MS", " ");
        htblStatesOfSoutheastRegion.put("NC", " ");
        htblStatesOfSoutheastRegion.put("PR", " ");
        htblStatesOfSoutheastRegion.put("SC", " ");
        htblStatesOfSoutheastRegion.put("TN", " ");
        htblStatesOfSoutheastRegion.put("VI", " ");

        htblStatesOfSouthwestRegion = new Hashtable();
        htblStatesOfSouthwestRegion.put("AZ", " ");
        htblStatesOfSouthwestRegion.put("CA", " ");
        htblStatesOfSouthwestRegion.put("CO", " ");
        htblStatesOfSouthwestRegion.put("GU", " ");
        htblStatesOfSouthwestRegion.put("HI", " ");
        htblStatesOfSouthwestRegion.put("NM", " ");
        htblStatesOfSouthwestRegion.put("NV", " ");
        htblStatesOfSouthwestRegion.put("OK", " ");
        htblStatesOfSouthwestRegion.put("TX", " ");
        htblStatesOfSouthwestRegion.put("UT", " ");

        htblStatesOfNorthwestRegion = new Hashtable();
        htblStatesOfNorthwestRegion.put("AK", " ");
        htblStatesOfNorthwestRegion.put("IA", " ");
        htblStatesOfNorthwestRegion.put("ID", " ");
        htblStatesOfNorthwestRegion.put("KS", " ");
        htblStatesOfNorthwestRegion.put("MN", " ");
        htblStatesOfNorthwestRegion.put("MO", " ");
        htblStatesOfNorthwestRegion.put("MT", " ");
        htblStatesOfNorthwestRegion.put("ND", " ");
        htblStatesOfNorthwestRegion.put("NE", " ");
        htblStatesOfNorthwestRegion.put("OR", " ");
        htblStatesOfNorthwestRegion.put("SD", " ");
        htblStatesOfNorthwestRegion.put("WA", " ");
        htblStatesOfNorthwestRegion.put("WY", " ");

        htblStatesOfNortheastRegion = new Hashtable();
        htblStatesOfNortheastRegion.put("CT", " ");
        htblStatesOfNortheastRegion.put("DC", " ");
        htblStatesOfNortheastRegion.put("DE", " ");
        htblStatesOfNortheastRegion.put("IL", " ");
        htblStatesOfNortheastRegion.put("IN", " ");
        htblStatesOfNortheastRegion.put("MA", " ");
        htblStatesOfNortheastRegion.put("MD", " ");
        htblStatesOfNortheastRegion.put("ME", " ");
        htblStatesOfNortheastRegion.put("MI", " ");
        htblStatesOfNortheastRegion.put("NH", " ");
        htblStatesOfNortheastRegion.put("NJ", " ");
        htblStatesOfNortheastRegion.put("NY", " ");
        htblStatesOfNortheastRegion.put("OH", " ");
        htblStatesOfNortheastRegion.put("PA", " ");
        htblStatesOfNortheastRegion.put("RI", " ");
        htblStatesOfNortheastRegion.put("VA", " ");
        htblStatesOfNortheastRegion.put("VT", " ");
        htblStatesOfNortheastRegion.put("WI", " ");
        htblStatesOfNortheastRegion.put("WV", " ");

        scrMainScreen.setVisible(true);

    }

    // this is a fancier border
    public void drawWindowBorder(Graphics g, int width, int height){
```

```
        // left border
        g.setColor(Color.lightGray);
        g.drawLine(0,0, 0, height-2);
        g.setColor(Color.white);
        g.drawLine(1, 0, 1, height-3);
        g.setColor(Color.lightGray);
        g.drawLine(2, 0, 2, height-3);
        g.drawLine(3, 0, 3, height-3);
        g.setColor(Color.gray);
        g.drawLine(4, 0, 4, height-6);
        g.setColor(Color.black);
        g.drawLine(5, 0, 5, height-7);

        // bottom border
        g.drawLine(0, height -1, width -1, height -1);
        g.setColor(Color.gray);
        g.drawLine(1, height-2, width -2, height -2);
        g.setColor(Color.lightGray);
        g.drawLine(4, height-3, width -5, height -3);
        g.drawLine(4, height -4, width -5, height -4);
        g.setColor(Color.white);
        g.drawLine(4, height -5, width - 5, height -5);
        g.setColor(Color.lightGray);
        g.drawLine(5, height -6, width - 6, height -6);

        // right border
        g.setColor(Color.black);
        g.drawLine(width -1, 0, width -1, height -1);
        g.setColor(Color.gray);
        g.drawLine(width -2, 1, width -2, height -2);
        g.setColor(Color.lightGray);
        g.drawLine(width -3, 3, width -3, height -3);
        g.drawLine(width -4, 3, width -4, height -3);
        g.setColor(Color.white);
        g.drawLine(width -5, 3, width -5, height -5);
        g.setColor(Color.lightGray);
        g.drawLine(width -6, 4, width -6, height -6);

        // top border
        g.setColor(Color.lightGray);
        g.drawLine(0, 0, width -2, 0);
        g.setColor(Color.white);
        g.drawLine(0, 1, width -3, 1);
        g.setColor(Color.lightGray);
        g.drawLine(2, 2, width -3, 2);
        g.drawLine(2, 3, width -3, 3);
        g.setColor(Color.gray);
        g.drawLine(4, 4, width -6, 4);
        g.setColor(Color.black);
        g.drawLine(5, 5, width -7, 5);
    } // end drawWindowBorder


    // this method does the transfer work to/from the server
    public java.lang.Object[] doServletRequest(java.lang.Object[] paramsToServer){
        Object[] dataTransportArray = null;
        ObjectOutputStream objectSender;
        ObjectInputStream objectReceiver;
        try{
            URL url = new URL("http", getCodeBase().getHost(), 80, "/servlet/DartServlet");
            URLConnection servletConnection = url.openConnection();

            servletConnection.setDoInput(true);
            servletConnection.setDoOutput(true);
            servletConnection.setUseCaches(false);
```

```
            servletConnection.setDefaultUseCaches(false);
            servletConnection.setRequestProperty("Content-Type", "java-internal/" + "[Ljava.lan
g.object;");
            objectSender = new ObjectOutputStream(servletConnection.getOutputStream());
            objectSender.writeObject(paramsToServer);
            objectSender.flush();
            objectReceiver = new ObjectInputStream(servletConnection.getInputStream());
            dataTransportArray = (java.lang.Object[])(objectReceiver.readObject());
            objectSender.close();
            objectReceiver.close();
        }
        catch(Exception e){
            System.out.println("Exception in gidb.doServletRequest");
            e.printStackTrace();
        }
        return dataTransportArray;
    }


    public void destroy(){
        wndNewLEAInformationWindow.dispose();
        wndLEAITCensusWindow.dispose();
        wndMapOverviewWindow.dispose();
        wndLEAStatisticsWindow.dispose();
    }


}
```

```
import java.io.*;
public class CDCRecord implements Serializable{
   public String username;
   public String service;
   public String rank;
   public String name;
   public String address;
   public String city;
   public String state;
   public String zip;
   public String lat;
   public String lon;
   public String voice;
   public String fax;
   public String DSNVoice;
   public String DSNFax;
   public String email;
   public String url;
   public String password;

   public String toString(){
      return "username: " + username + " service:" + service + " rank:" + rank + " name:" +
name + " address:" + address + " city:" + city +
            " state:" + state + " zip:" + zip + " lat:" + lat + " lon:" + lon + " voice:" +
 voice +
            " fax:" + fax + " DSNVoice:" + DSNVoice + " DSNFax:" + DSNFax + " Email: " + em
ail + " URL: " + url;
   }

}
```

```
import java.io.*;
public class LEARecord implements Serializable{
   public String username;
   public String jurisdictionType;
   public String jurisdictionName;
   public String rank;
   public String name;
   public String CDCUsername;
   public String address;
   public String city;
   public String state;
   public String zip;
   public String lat;
   public String lon;
   public String voice;
   public String fax;
   public String DSNVoice;
   public String DSNFax;
   public String email;
   public String url;
   public String password;
   public String strAccessToComputer;
   public String strTypeOfComputer;
   public String strSpeedOfComputer;
   public String strOperatingSystem;
   public String strConnectedToNetwork;
   public String strTypeOfNetworkConnection;
   public String strSpeedOfNetworkConnection;
   public String strBrowser;
   public String strTypeOfBrowser;
   public String strViewedWebPage;

   public String toString(){
      return "username: " + username + " jurisdictionType:" + jurisdictionType + " jurisdict
ionName: " + jurisdictionName +
             " rank:" + rank + " name:" + name + " CDC: " + CDCUsername + " address:" + addr
ess + " city:" + city +
             " state:" + state + " zip:" + zip + " lat:" + lat + " lon:" + lon + " voice:" +
 voice +
             " fax:" + fax + " DSNVoice:" + DSNVoice + " DSNFax:" + DSNFax + " Email: " + em
ail + " URL: " + url;
   }

}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.io.*;
public class DartAppletCDCInfoScreen extends Panel implements ActionListener, ItemListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   MouseOverButton btnBack, btnNextPage, btnHome, btnEdit, btnSave, btnCancel;
   Label lblState, lblCurrentCDCoordinator, lblCDCJurisdiction, lblAddress, lblPOCLocation,
lblLatitude,
         lblLongitude, lblRankTitle, lblCity, lblContactState, lblZip, lblCommVoice, lblComm
Fax, lblDSNVoice,
         lblDSNFax, lblEmail, lblURLWWWIntranet, lblUsername, lblPassword;
   List lstStates;
   TextArea txtAddress;
   TextField txtCurrentCDCoordinator, txtCDCJurisdiction, txtLatitude, txtLongitude,
            txtRankTitle, txtCity, txtContactState, txtZip, txtCommVoice, txtCommFax, txtDS
NVoice, txtDSNFax,
            txtEmail, txtURLWWWIntranet, txtUsername, txtPassword;
   Vector vtrCDCRecords;

   Image tempMapImage;

   ConfirmEditWindow editConfirmationWindow;

   CDCRecord currentRecord;

   public DartAppletCDCInfoScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);

      lblState = new AutoLabel("State:", mainAppletContext.regularScreenFont);
      lblState.setLocation(50, 120);
      add(lblState);

      lstStates = new List();
      lstStates.setSize(50, 80);
      lstStates.setLocation(50, 140);
      lstStates.addItemListener(this);
      add(lstStates);

      lblCurrentCDCoordinator = new AutoLabel("Current CD Coordinator:", mainAppletContext.r
egularScreenFont);
      lblCurrentCDCoordinator.setLocation(120, 120);
      add(lblCurrentCDCoordinator);

      txtCurrentCDCoordinator = new TextField("");
      txtCurrentCDCoordinator.setSize(160, 20);
      txtCurrentCDCoordinator.setLocation(120, 140);
      add(txtCurrentCDCoordinator);

      lblCDCJurisdiction = new AutoLabel("CDC Jurisdiction:", mainAppletContext.regularScree
nFont);
      lblCDCJurisdiction.setLocation(120, 170);
      add(lblCDCJurisdiction);

      txtCDCJurisdiction = new TextField("");
      txtCDCJurisdiction.setSize(160, 20);
      txtCDCJurisdiction.setLocation(120, 190);
      add(txtCDCJurisdiction);

      lblAddress = new AutoLabel("Address:", mainAppletContext.regularScreenFont);
```

```
        lblAddress.setLocation(320, 120);
        add(lblAddress);

        txtAddress = new TextArea("");
        txtAddress.setSize(260, 60);
        txtAddress.setLocation(320, 140);
        add(txtAddress);

        lblPOCLocation = new AutoLabel("POC Location", mainAppletContext.regularScreenFont);
        lblPOCLocation.setLocation(580, 120);
        //add(lblPOCLocation);

        lblCity = new AutoLabel("City:", mainAppletContext.regularScreenFont);
        lblCity.setLocation(50, 250);
        add(lblCity);

        txtCity = new TextField("");
        txtCity.setSize(100, 20);
        txtCity.setLocation(50, 270);
        add(txtCity);

        lblContactState = new AutoLabel("State:", mainAppletContext.regularScreenFont);
        lblContactState.setLocation(170, 250);
        add(lblContactState);

        txtContactState = new TextField("");
        txtContactState.setSize(100, 20);
        txtContactState.setLocation(170, 270);
        add(txtContactState);

        lblZip = new AutoLabel("Zip:", mainAppletContext.regularScreenFont);
        lblZip.setLocation(50, 300);
        add(lblZip);

        txtZip = new TextField("");
        txtZip.setSize(100, 20);
        txtZip.setLocation(50, 320);
        add(txtZip);

        lblRankTitle = new AutoLabel("Rank/Title:", mainAppletContext.regularScreenFont);
        lblRankTitle.setLocation(170, 300);
        add(lblRankTitle);

        txtRankTitle = new TextField("");
        txtRankTitle.setSize(100, 20);
        txtRankTitle.setLocation(170, 320);
        add(txtRankTitle);

        lblCommVoice = new AutoLabel("Comm. Voice:", mainAppletContext.regularScreenFont);
        lblCommVoice.setLocation(50, 350);
        add(lblCommVoice);

        txtCommVoice = new TextField("");
        txtCommVoice.setSize(100, 20);
        txtCommVoice.setLocation(50, 370);
        add(txtCommVoice);

        lblCommFax = new AutoLabel("Comm. Fax:", mainAppletContext.regularScreenFont);
        lblCommFax.setLocation(170, 350);
        add(lblCommFax);

        txtCommFax = new TextField("");
        txtCommFax.setSize(100, 20);
        txtCommFax.setLocation(170, 370);
        add(txtCommFax);
```

```
        lblDSNVoice = new AutoLabel("DSN, Voice:", mainAppletContext.regularScreenFont);
        lblDSNVoice.setLocation(50, 400);
        add(lblDSNVoice);

        txtDSNVoice = new TextField("");
        txtDSNVoice.setSize(100, 20);
        txtDSNVoice.setLocation(50, 420);
        add(txtDSNVoice);

        lblDSNFax = new AutoLabel("DSN, Fax:", mainAppletContext.regularScreenFont);
        lblDSNFax.setLocation(170, 400);
        add(lblDSNFax);

        txtDSNFax = new TextField("");
        txtDSNFax.setSize(100, 20);
        txtDSNFax.setLocation(170, 420);
        add(txtDSNFax);

        lblEmail = new AutoLabel("Email:", mainAppletContext.regularScreenFont);
        lblEmail.setLocation(50, 450);
        add(lblEmail);

        txtEmail = new TextField("");
        txtEmail.setSize(100, 20);
        txtEmail.setLocation(50, 470);
        add(txtEmail);

        lblURLWWWIntranet = new AutoLabel("URL:", mainAppletContext.regularScreenFont);
        lblURLWWWIntranet.setLocation(170, 450);
        add(lblURLWWWIntranet);

        txtURLWWWIntranet = new TextField("");
        txtURLWWWIntranet.setSize(100, 20);
        txtURLWWWIntranet.setLocation(170, 470);
        add(txtURLWWWIntranet);

        lblUsername = new AutoLabel("Username:", mainAppletContext.regularScreenFont);
        lblUsername.setLocation(50, 500);
        add(lblUsername);

        txtUsername = new TextField("");
        txtUsername.setSize(100, 20);
        txtUsername.setLocation(50, 520);
        add(txtUsername);

        lblPassword = new AutoLabel("Password:", mainAppletContext.regularScreenFont);
        lblPassword.setLocation(170, 500);
        add(lblPassword);

        txtPassword = new TextField("");
        txtPassword.setSize(100, 20);
        txtPassword.setLocation(170, 520);
        add(txtPassword);

        btnBack = new MouseOverButton("Back");
        btnBack.setSize(80, 20);
        btnBack.setLocation(320, 540);
        btnBack.addActionListener(this);
        btnBack.setFont(mainAppletContext.regularScreenFont);
        add(btnBack);

        btnNextPage = new MouseOverButton("Next Page");
        btnNextPage.setSize(80, 20);
        btnNextPage.setLocation(410, 540);
```

```
        btnNextPage.addActionListener(this);
        btnNextPage.setFont(mainAppletContext.regularScreenFont);
        add(btnNextPage);

        btnHome = new MouseOverButton("Home");
        btnHome.setSize(80, 20);
        btnHome.setLocation(500, 540);
        btnHome.addActionListener(this);
        btnHome.setFont(mainAppletContext.regularScreenFont);
        add(btnHome);

        lblLatitude = new AutoLabel("Lat:", mainAppletContext.regularScreenFont);
        lblLatitude.setLocation(700, 400);
        add(lblLatitude);

        txtLatitude = new TextField("");
        txtLatitude.setSize(50, 20);
        txtLatitude.setLocation(700, 420);
        add(txtLatitude);

        lblLongitude = new AutoLabel("Lon:", mainAppletContext.regularScreenFont);
        lblLongitude.setLocation(700, 450);
        add(lblLongitude);

        txtLongitude = new TextField("");
        txtLongitude.setSize(50, 20);
        txtLongitude.setLocation(700, 470);
        add(txtLongitude);

        btnEdit = new MouseOverButton("Edit");
        btnEdit.setSize(60, 20);
        btnEdit.setLocation(630, 140);
        btnEdit.addActionListener(this);
        btnEdit.setVisible(false);
        add(btnEdit);

        btnCancel = new MouseOverButton("Cancel");
        btnCancel.setSize(60, 20);
        btnCancel.setLocation(630, 140);
        btnCancel.addActionListener(this);
        btnCancel.setVisible(false);
        add(btnCancel);

        btnSave = new MouseOverButton("Save");
        btnSave.setSize(60, 20);
        btnSave.setLocation(630, 170);
        btnSave.addActionListener(this);
        btnSave.setVisible(false);
        add(btnSave);

        editConfirmationWindow = new ConfirmEditWindow();

    }

    public void setEditableState(boolean editableState){
        txtCurrentCDCoordinator.setEditable(editableState);
        txtCDCJurisdiction.setEditable(editableState);
        txtAddress.setEditable(editableState);
        txtLatitude.setEditable(editableState);
        txtLongitude.setEditable(editableState);
        txtRankTitle.setEditable(editableState);
        txtCity.setEditable(editableState);
        txtContactState.setEditable(editableState);
        txtZip.setEditable(editableState);
        txtCommVoice.setEditable(editableState);
```

```
            txtUsername.setVisible(false);
            lblPassword.setVisible(false);
            txtPassword.setVisible(false);
            btnEdit.setVisible(false);
            btnCancel.setVisible(false);
            btnSave.setVisible(false);
         }
         setFields();
         RetrieveCDCImageThread rcit = new RetrieveCDCImageThread(this);
         rcit.start();
      }

      class RetrieveCDCImageThread extends Thread{
         Component componentForMediaTracker;
         RetrieveCDCImageThread(Component componentForMediaTracker){
            this.componentForMediaTracker = componentForMediaTracker;
         }
         public void run(){
            try{
               if (tempMapImage != null){
                  tempMapImage.flush();
                  tempMapImage = null;
               }
               repaint();

               Object[] dataTransportArray = new Object[8];
               dataTransportArray[0] = "UserImageRequest";
               dataTransportArray[1] = "CDCView";
               dataTransportArray[2] = currentRecord.name;
               dataTransportArray[3] = currentRecord.lat;
               dataTransportArray[4] = currentRecord.lon;
               dataTransportArray[5] = currentRecord.state;
               dataTransportArray[6] = "State";
               dataTransportArray[7] = currentRecord.state;
               dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);

               String filePrefix = (String)dataTransportArray[0];
               String imageFileName = filePrefix + ".jpg";
               String fileFinishedMarkerFileName = filePrefix + ".finished";
               URL finishedFileURL = new URL(mainAppletContext.getCodeBase() + fileFinishedMark
erFileName);
               boolean fileCreated = false;
               int numberOfTries = 0;
               InputStream tempIS = null;
               while ( (! fileCreated) && (numberOfTries < 100) ){
                  try{
                     sleep(200);
                     tempIS = finishedFileURL.openStream();
                     tempIS.close();
                     fileCreated = true;
                  }
                  catch(Exception fileNotYetThereException){
                     numberOfTries++;
                     if (numberOfTries == 100){
                        System.out.println("CDC image file not found");
                     }
                  }
               }
               if (fileCreated){
                  Image unscaledTempMapImage = mainAppletContext.getImage(mainAppletContext.get
CodeBase(), imageFileName);
                  MediaTracker mt = new MediaTracker(componentForMediaTracker);
                  mt.addImage(unscaledTempMapImage, 0);
                  mt.waitForAll();
                  mt.removeImage(unscaledTempMapImage, 0);
```

```
      }
      super.setVisible(visible);
      if (visible){
         for (int i = 0; i < vtrCDCRecords.size(); i++){
            lstStates.addItem( ((CDCRecord)(vtrCDCRecords.elementAt(i))).state );
         }
         int currentRecordIndex = vtrCDCRecords.indexOf(currentRecord);
         lstStates.select(currentRecordIndex);
         lstStates.makeVisible(currentRecordIndex);
         showRecord(currentRecordIndex);
      }
      if ( visible && mainAppletContext.blnInNewLEAMode){
         mainAppletContext.wndNewLEAInformationWindow.setMessage("This is the information fo
r the state you " +
                                                                 "have selected and the Cou
nter-Drug Coordinator " +
                                                                 "(CDC) for your region.  I
f this information appears " +
                                                                 "correct, press \"Next Pag
e\" to continue.  If not, " +
                                                                 "select another state from
 the list of states.");
         mainAppletContext.wndNewLEAInformationWindow.show();

         btnHome.setLabel("Cancel");
      }
   }

   public void setCDCRecords(Vector vtrCDCRecords){
      this.vtrCDCRecords = vtrCDCRecords;
   }

   public void setSelectedRecord(int recordIndex){
      currentRecord = (CDCRecord)vtrCDCRecords.elementAt(recordIndex);
   }

   public void disableNonEditActionComponents(){
      btnBack.setEnabled(false);
      btnNextPage.setEnabled(false);
      btnHome.setEnabled(false);
      lstStates.setEnabled(false);
      mainAppletContext.navigationPanel.disableCDCButton();
      mainAppletContext.navigationPanel.disableLEAButton();
      mainAppletContext.navigationPanel.disableHardwareButton();
      mainAppletContext.navigationPanel.disableOSSoftwareButton();
      mainAppletContext.navigationPanel.disableNetworkButton();
      mainAppletContext.navigationPanel.disableWebConnectButton();
   }

   public void enableNonEditActionComponents(){
      btnBack.setEnabled(true);
      btnNextPage.setEnabled(true);
      btnHome.setEnabled(true);
      lstStates.setEnabled(true);
      mainAppletContext.navigationPanel.enableCDCButton();
      mainAppletContext.navigationPanel.enableLEAButton();
      mainAppletContext.navigationPanel.enableHardwareButton();
      mainAppletContext.navigationPanel.enableOSSoftwareButton();
      mainAppletContext.navigationPanel.enableNetworkButton();
      mainAppletContext.navigationPanel.enableWebConnectButton();
   }

   public void btnBackClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrITCensusScreen1);
   }
```

```
    public void btnNextPageClicked(){
        mainAppletContext.scrLEAInfoScreen.setCDC( currentRecord );
        mainAppletContext.showScreen(mainAppletContext.scrLEAInfoScreen);


    public void btnHomeClicked(){
        mainAppletContext.showScreen(mainAppletContext.scrMainScreen);


    public void btnEditClicked(){
        btnEdit.setVisible(false);
        btnCancel.setVisible(true);
        btnSave.setVisible(true);
        setEditableState(true);
        disableNonEditActionComponents();


    public void btnCancelClicked(){
        btnCancel.setVisible(false);
        btnSave.setVisible(false);
        btnEdit.setVisible(true);
        setEditableState(false);
        setFields();   // this will set the textboxes back to the original value
        enableNonEditActionComponents();


    public void btnSaveClicked(){
        editConfirmationWindow.show();


    public void actionPerformed(ActionEvent ae){
        Object eventSource = ae.getSource();
        if (eventSource == btnBack){
            btnBackClicked();

        else if (eventSource == btnNextPage){
            btnNextPageClicked();

        else if (eventSource == btnHome){
            btnHomeClicked();

        else if (eventSource == btnEdit){
            btnEditClicked();

        else if (eventSource == btnCancel){
            btnCancelClicked();

        else if (eventSource == btnSave){
            btnSaveClicked();
        }
    }

    public void lstStatesClicked(){
        showRecord(lstStates.getSelectedIndex());
        currentRecord = (CDCRecord)vtrCDCRecords.elementAt(lstStates.getSelectedIndex());
    }

    public void itemStateChanged(ItemEvent ie){
        Object eventSource = ie.getSource();
        if (eventSource == lstStates){
            lstStatesClicked();
        }
    }
```

```
    public void paint(Graphics g){
        g.setFont(mainAppletContext.screenTitleFont);
        fm = g.getFontMetrics();

        tempString = "CDC Info";
        g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

        if (tempMapImage != null){
            g.drawImage(tempMapImage, 350, 230, null);
        }

        g.drawRect(349, 229, 343, 301);

        mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
    }

    public void updateRecordAndSendToServer(){
        currentRecord.service = txtCDCJurisdiction.getText();
        currentRecord.rank = txtRankTitle.getText();
        currentRecord.name = txtCurrentCDCoordinator.getText();
        currentRecord.address = txtAddress.getText();
        currentRecord.city = txtCity.getText();
        currentRecord.state = txtContactState.getText();
        currentRecord.zip = txtZip.getText();
        currentRecord.lat = txtLatitude.getText();
        currentRecord.lon = txtLongitude.getText();
        currentRecord.voice = txtCommVoice.getText();
        currentRecord.fax = txtCommFax.getText();
        currentRecord.DSNVoice = txtDSNVoice.getText();
        currentRecord.DSNFax = txtDSNFax.getText();
        currentRecord.email = txtEmail.getText();
        currentRecord.url = txtURLWWWIntranet.getText();

        if (currentRecord.service.equals("")){
            currentRecord.service = " ";
        }
        if (currentRecord.rank.equals("")){
            currentRecord.rank = " ";
        }
        if (currentRecord.name.equals("")){
            currentRecord.name = " ";
        }
        if (currentRecord.address.equals("")){
            currentRecord.address = " ";
        }
        if (currentRecord.city.equals("")){
            currentRecord.city = " ";
        }
        if (currentRecord.state.equals("")){
            currentRecord.state = " ";
        }
        if (currentRecord.zip.equals("")){
            currentRecord.zip = " ";
        }
        if (currentRecord.lat.equals("")){
            currentRecord.lat = " ";
        }
        if (currentRecord.lon.equals("")){
            currentRecord.lon = " ";
        }
        if (currentRecord.voice.equals("")){
            currentRecord.voice = " ";
        }
        if (currentRecord.fax.equals("")){
```

```
            currentRecord.fax = " ";
         }
         if (currentRecord.DSNVoice.equals("")){
            currentRecord.DSNVoice = " ";
         }
         if (currentRecord.DSNFax.equals("")){
            currentRecord.DSNFax = " ";
         }
         if (currentRecord.email.equals("")){
            currentRecord.email = " ";
         }
         if (currentRecord.url.equals("")){
            currentRecord.url = " ";
         }

         Object[] dataTransportArray = new Object[8];
         dataTransportArray[0] = "updateCDCRecord";
         dataTransportArray[1] = currentRecord;
         dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
         enableNonEditActionComponents();
      }


      class ConfirmEditWindow extends Frame implements ActionListener, WindowListener{
         MouseOverButton btnYes;
         MouseOverButton btnNo;
         AutoLabel lblConfirmMessage;
         ConfirmEditWindow(){
            setBackground(Color.lightGray);
            setResizable(false);
            setLayout(null);
            setSize(350, 170);
            setLocation(300, 300);
            setTitle("Save Changes Confirmation");

            lblConfirmMessage = new AutoLabel("Do you really want to save changes?", mainApplet
Context.regularScreenFont);
            lblConfirmMessage.setLocation(50, 60);
            add(lblConfirmMessage);

            btnYes = new MouseOverButton("Yes");
            btnYes.setSize(60, 20);
            btnYes.setLocation(100, 110);
            btnYes.addActionListener(this);
            add(btnYes);

            btnNo = new MouseOverButton("No");
            btnNo.setSize(60, 20);
            btnNo.setLocation(180, 110);
            btnNo.addActionListener(this);
            add(btnNo);

            addWindowListener(this);
         }
         public void btnYesClicked(){
            // send update to servlet
            updateRecordAndSendToServer();

            setEditableState(false);
            btnCancel.setVisible(false);
            btnSave.setVisible(false);
            btnEdit.setVisible(true);
            this.setVisible(false);
         }
         public void btnNoClicked(){
```

```
            this.setVisible(false);
        }
        public void actionPerformed(ActionEvent ae){
          Object eventSource = ae.getSource();
          if (eventSource == btnYes){
             btnYesClicked();
          }
          else if (eventSource == btnNo){
             btnNoClicked();
          }
        }

        public void windowActivated(WindowEvent we){}
        public void windowClosed(WindowEvent we){}
        public void windowClosing(WindowEvent we){
           this.setVisible(false);
        }
        public void windowDeactivated(WindowEvent we){
           this.setVisible(false);
        }
        public void windowIconified(WindowEvent we){}
        public void windowDeiconified(WindowEvent we){}
        public void windowOpened(WindowEvent we){}

    }

}
```

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;

public class DartAppletITCensusScreen1 extends Panel implements ActionListener, MouseListene
r, MouseMotionListener, ItemListener{
   DartApplet mainAppletContext;
   Image regionMapImage = null;
   String tempString;

   FontMetrics fm;

   Label lblTitle;
   Label lblSelectedRegion;
   Label lblStateOrTerritory;
   Label lblCDC;

   Polygon plgSouthwestOutline;
   Polygon plgSoutheastOutline;
   Polygon plgNorthwestOutline;
   Polygon plgNortheastOutline;

   MapCanvas theMap;

   Vector vtrCDCRecords;

   List lstStates;
   TextField txtCDCName;
   MouseOverButton btnBack, btnNextPage, btnHome;

   int currentlySelectedIndex;

   public DartAppletITCensusScreen1(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);
      MediaTracker mt = new MediaTracker(this);
      regionMapImage = mainAppletContext.getImage(mainAppletContext.getDocumentBase(), "Regi
onMap.jpg");
      mt.addImage(regionMapImage, 0);

      try{
         mt.waitForAll();
      }
      catch(Exception e){
         e.printStackTrace();
      }

      lblTitle = new AutoLabel("CDC SELECTION - Select region and state", mainAppletContext.
screenTitleFont);
      lblTitle.setLocation( (int)(mainAppletContext.getSize().width/2.0 - lblTitle.getSize()
.width/2.0 + 0.5), 10);
      add(lblTitle);

      lblSelectedRegion = new Label("Selected Region: None");
      lblSelectedRegion.setFont(mainAppletContext.regularScreenFont);
      lblSelectedRegion.setSize(180, 20);
      lblSelectedRegion.setLocation(30, 200);
      add(lblSelectedRegion);

      plgSouthwestOutline = new Polygon();
      plgSouthwestOutline.addPoint(230, 272);
      plgSouthwestOutline.addPoint(51, 240);
```

```
        plgSouthwestOutline.addPoint(18, 200);
        plgSouthwestOutline.addPoint(46, 129);
        plgSouthwestOutline.addPoint(145, 132);
        plgSouthwestOutline.addPoint(146, 141);
        plgSouthwestOutline.addPoint(201, 141);
        plgSouthwestOutline.addPoint(202, 174);
        plgSouthwestOutline.addPoint(255, 176);
        plgSouthwestOutline.addPoint(255, 239);

        plgSoutheastOutline = new Polygon();
        plgSoutheastOutline.addPoint(275, 286);
        plgSoutheastOutline.addPoint(385, 332);
        plgSoutheastOutline.addPoint(461, 286);
        plgSoutheastOutline.addPoint(432, 207);
        plgSoutheastOutline.addPoint(385, 206);
        plgSoutheastOutline.addPoint(371, 187);
        plgSoutheastOutline.addPoint(355, 182);
        plgSoutheastOutline.addPoint(327, 194);
        plgSoutheastOutline.addPoint(268, 209);

        plgNorthwestOutline = new Polygon();
        plgNorthwestOutline.addPoint(10, 8);
        plgNorthwestOutline.addPoint(250, 26);
        plgNorthwestOutline.addPoint(291, 47);
        plgNorthwestOutline.addPoint(279, 79);
        plgNorthwestOutline.addPoint(296, 118);
        plgNorthwestOutline.addPoint(306, 175);
        plgNorthwestOutline.addPoint(207, 167);
        plgNorthwestOutline.addPoint(205, 134);
        plgNorthwestOutline.addPoint(140, 125);
        plgNorthwestOutline.addPoint(9, 70);

        plgNortheastOutline = new Polygon();
        plgNortheastOutline.addPoint(521, 28);
        plgNortheastOutline.addPoint(522, 88);
        plgNortheastOutline.addPoint(446, 193);
        plgNortheastOutline.addPoint(385, 191);
        plgNortheastOutline.addPoint(360, 173);
        plgNortheastOutline.addPoint(330, 188);
        plgNortheastOutline.addPoint(307, 140);
        plgNortheastOutline.addPoint(308, 98);
        plgNortheastOutline.addPoint(292, 68);
        plgNortheastOutline.addPoint(315, 37);

        addMouseMotionListener(this);

        theMap = new MapCanvas(regionMapImage);
        theMap.setSize(regionMapImage.getWidth(null), regionMapImage.getHeight(null));
        //theMap.setLocation(122, 100);
        theMap.setLocation(222, 180);
        add(theMap);

        lblStateOrTerritory = new AutoLabel("State/Territory:", mainAppletContext.regularScree
nFont);
        lblStateOrTerritory.setLocation(30, 250);
        add(lblStateOrTerritory);

        lstStates = new List();
        lstStates.setSize(50, 80);
        lstStates.setLocation(30, 270);
        lstStates.addItemListener(this);
        add(lstStates);

        lblCDC = new AutoLabel("CDC:", mainAppletContext.regularScreenFont);
        lblCDC.setLocation(30, 370);
```

```
      add(lblCDC);

      txtCDCName =  new TextField("");
      txtCDCName.setSize(180, 20);
      txtCDCName.setLocation(30, 390);
      txtCDCName.setEditable(false);
      txtCDCName.setBackground(Color.white);
      add(txtCDCName);

      btnBack = new MouseOverButton("Back");
      btnBack.setSize(80, 20);
      btnBack.setLocation(320, 540);
      btnBack.addActionListener(this);
      btnBack.setFont(mainAppletContext.regularScreenFont);
      add(btnBack);

      btnNextPage = new MouseOverButton("Next Page");
      btnNextPage.setSize(80, 20);
      btnNextPage.setLocation(410, 540);
      btnNextPage.addActionListener(this);
      btnNextPage.setFont(mainAppletContext.regularScreenFont);
      add(btnNextPage);

      btnHome = new MouseOverButton("Home");
      btnHome.setSize(80, 20);
      btnHome.setLocation(500, 540);
      btnHome.addActionListener(this);
      btnHome.setFont(mainAppletContext.regularScreenFont);
      add(btnHome);

   }

   public int getIndexOfState(String state){
      String[] states = lstStates.getItems();
      int index = -1;
      for (int i = 0; i < states.length; i++){
         if (states[i].equals(state)){
            index = i;
            break;
         }
      }
      return index;
   }

   public void setVisible(boolean visible){
      if (visible){

         if (mainAppletContext.fpNextItemPrompt.getParent() != null){
            mainAppletContext.fpNextItemPrompt.getParent().remove(mainAppletContext.fpNextIt
emPrompt);
         }

         mainAppletContext.navigationPanel.resetButtons();
         mainAppletContext.navigationPanel.setVisible(true);
         mainAppletContext.navigationPanel.disableLEAButton();
         lblSelectedRegion.setText("Selected Region: None");
         lstStates.removeAll();
         txtCDCName.setText("");
         btnNextPage.setEnabled(false);
         if (! mainAppletContext.blnInNewLEAMode){
            if (mainAppletContext.htblStatesOfSoutheastRegion.containsKey(mainAppletContext.
strCurrentLoggedInUserState)){
               retrieveCDCsForRegion("Southeast");
               int indexOfState = getIndexOfState(mainAppletContext.strCurrentLoggedInUserSt
ate);
```

```
                lstStates.select(indexOfState);
                lstStates.makeVisible(indexOfState);
                lstStatesClicked();
             }
             else if (mainAppletContext.htblStatesOfSouthwestRegion.containsKey(mainAppletCon
text.strCurrentLoggedInUserState)){
                retrieveCDCsForRegion("Southwest");
                int indexOfState = getIndexOfState(mainAppletContext.strCurrentLoggedInUserSt
ate);
                lstStates.select(indexOfState);
                lstStates.makeVisible(indexOfState);
                lstStatesClicked();
             }
             else if (mainAppletContext.htblStatesOfNorthwestRegion.containsKey(mainAppletCon
text.strCurrentLoggedInUserState)){
                retrieveCDCsForRegion("Northwest");
                int indexOfState = getIndexOfState(mainAppletContext.strCurrentLoggedInUserSt
ate);
                lstStates.select(indexOfState);
                lstStates.makeVisible(indexOfState);
                lstStatesClicked();
             }
             else if (mainAppletContext.htblStatesOfNortheastRegion.containsKey(mainAppletCon
text.strCurrentLoggedInUserState)){
                retrieveCDCsForRegion("Northeast");
                int indexOfState = getIndexOfState(mainAppletContext.strCurrentLoggedInUserSt
ate);
                lstStates.select(indexOfState);
                lstStates.makeVisible(indexOfState);
                lstStatesClicked();
             }
          }
       }
       super.setVisible(visible);
       if ( visible && mainAppletContext.blnInNewLEAMode){
          mainAppletContext.wndNewLEAInformationWindow.setMessage("Use the map to select your
 region of operations.  " +
                                                                  "Then select your state fro
m the List.  Select \"Next Page\" " +
                                                                  "when you are ready to cont
inue.");
          mainAppletContext.wndNewLEAInformationWindow.show();
          mainAppletContext.fpNextItemPrompt.setDirection("down");
          mainAppletContext.fpNextItemPrompt.setLocation(theMap.getLocation().x + 230, theMap
.getLocation().y-20);
          add(mainAppletContext.fpNextItemPrompt);
       }
    }

    public void paint(Graphics g){
       g.setFont(mainAppletContext.screenTitleFont);
       fm = g.getFontMetrics();

       mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
    }


    public void btnBackClicked(){
       mainAppletContext.showScreen(mainAppletContext.scrMainScreen);
    }

    public void btnNextPageClicked(){
       mainAppletContext.navigationPanel.viewCDCInfoClicked();
    }
```

```
    public void btnHomeClicked(){
        mainAppletContext.showScreen(mainAppletContext.scrMainScreen);
    }

    public void btnCDCInfoClicked(){
        PopupMenu CDCMenu = new PopupMenu();
        MenuItem firstItem = new MenuItem("View Info");
        firstItem.addActionListener(this);
        MenuItem secondItem = new MenuItem("Edit Info");
        secondItem.addActionListener(this);
        MenuItem thirdItem = new MenuItem("Email CDC");
        thirdItem.addActionListener(this);
        CDCMenu.add(firstItem);
        CDCMenu.add(secondItem);
        CDCMenu.add(thirdItem);
        add(CDCMenu);
        CDCMenu.show(this, 50, 130);
    }

    public void miViewInfoClicked(){
        if (! txtCDCName.getText().equals("")){
            setVisible(false);
            mainAppletContext.scrCDCInfoScreen.setCDCRecords(vtrCDCRecords);
            mainAppletContext.scrCDCInfoScreen.setSelectedRecord(currentlySelectedIndex);
            mainAppletContext.scrCDCInfoScreen.setVisible(true);
        }
    }

    public void miEmailCDCClicked(){
        try{
            mainAppletContext.getAppletContext().showDocument(new URL("mailto:" + ( ((CDCRecord
)(vtrCDCRecords.elementAt(
                                                    lstStates.getSelectedIndex()))).e
mail )));
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    public void actionPerformed(ActionEvent ae){
        String ac = ae.getActionCommand();
        if (ac.equals("Back")){
            btnBackClicked();
        }
        else if (ac.equals("Next Page")){
            btnNextPageClicked();
        }
        else if (ac.equals("Home")){
            btnHomeClicked();
        }
        else if (ac.equals("CDC Info")){
            btnCDCInfoClicked();
        }
        else if (ac.equals("View Info")){
            miViewInfoClicked();
        }
        else if (ac.equals("Email CDC")){
            miEmailCDCClicked();
        }
    }

    public void lstStatesClicked(){
        currentlySelectedIndex = lstStates.getSelectedIndex();
        txtCDCName.setText( ((CDCRecord)(vtrCDCRecords.elementAt(lstStates.getSelectedIndex()
```

```
)).name );
      btnNextPage.setEnabled(true);
      mainAppletContext.navigationPanel.enableCDCButton();

      if (mainAppletContext.fpNextItemPrompt.getParent() != null){
         mainAppletContext.fpNextItemPrompt.getParent().remove(mainAppletContext.fpNextItemP
rompt);
      }
      if (mainAppletContext.blnInNewLEAMode){
         mainAppletContext.fpNextItemPrompt.setDirection("down");
         mainAppletContext.fpNextItemPrompt.setLocation(btnNextPage.getLocation().x, btnNext
Page.getLocation().y-20);
         add(mainAppletContext.fpNextItemPrompt);
      }


   }

   public void itemStateChanged(ItemEvent ie){
      lstStatesClicked();
   }

   public void retrieveCDCsForRegion(String strRegion){
      Object[] dataTransportArray = new Object[1];
      lstStates.removeAll();
      lblSelectedRegion.setText("Selected Region: " + strRegion);
      dataTransportArray[0] = "get" + strRegion + "CDCData";
      dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
      vtrCDCRecords = (Vector)(dataTransportArray[0]);
      for (int i = 0; i < vtrCDCRecords.size(); i++){
         lstStates.addItem( ((CDCRecord)(vtrCDCRecords.elementAt(i))).state);
      }
   }

   public void mouseEntered(MouseEvent me){}
   public void mouseExited(MouseEvent me){}
   public void mousePressed(MouseEvent me){}
   public void mouseClicked(MouseEvent me){}
   public void mouseReleased(MouseEvent me){}
   public void mouseDragged(MouseEvent me){}
   public void mouseMoved(MouseEvent me){    }

   public class MapCanvas extends Canvas implements MouseMotionListener, MouseListener{
      Image buffer;
      Graphics bufferG;
      Image mapImage;
      int mouseX, mouseY;
      Rectangle clipRectangle;
      boolean blnMouseInSouthwestArea = false;
      boolean blnMouseInSoutheastArea = false;
      boolean blnMouseInNorthwestArea = false;
      boolean blnMouseInNortheastArea = false;
      int currentState = 0;
      int lastState = 0;

      public MapCanvas(Image mapToShow){
         mapImage = mapToShow;
         addMouseMotionListener(this);
         addMouseListener(this);
      }

      public void paint(Graphics g){
         if (buffer == null){
            buffer = createImage(getSize().width, getSize().height);
            bufferG = buffer.getGraphics();
```

```
        }
        bufferG.drawImage(mapImage, 0, 0, null);
        bufferG.setColor(Color.blue);
        if (blnMouseInSouthwestArea){
           bufferG.drawPolygon(plgSouthwestOutline);
        }
        else if (blnMouseInSoutheastArea){
           bufferG.drawPolygon(plgSoutheastOutline);
        }
        else if (blnMouseInNorthwestArea){
           bufferG.drawPolygon(plgNorthwestOutline);
        }
        else if (blnMouseInNortheastArea){
           bufferG.drawPolygon(plgNortheastOutline);
        }
        bufferG.setColor(Color.black);
        bufferG.drawRect(0, 0, getSize().width -1, getSize().height -1);
        g.drawImage(buffer, 0, 0, null);
     }
     public void update(Graphics g){
        paint(g);
     }

     public void mouseEntered(MouseEvent me){}
     public void mouseExited(MouseEvent me){}
     public void mousePressed(MouseEvent me){}
     public void mouseClicked(MouseEvent me){
        txtCDCName.setText("");
        btnNextPage.setEnabled(false);
        mainAppletContext.navigationPanel.disableCDCButton();
        mainAppletContext.navigationPanel.disableLEAButton();
        if (blnMouseInSouthwestArea){
           retrieveCDCsForRegion("Southwest");
           mainAppletContext.fpNextItemPrompt.setDirection("right");
           mainAppletContext.fpNextItemPrompt.setLocation(lstStates.getLocation().x-20, lst
States.getLocation().y);
        }
        else if (blnMouseInSoutheastArea){
           retrieveCDCsForRegion("Southeast");
           mainAppletContext.fpNextItemPrompt.setDirection("right");
           mainAppletContext.fpNextItemPrompt.setLocation(lstStates.getLocation().x-20, lst
States.getLocation().y);
        }
        else if (blnMouseInNorthwestArea){
           retrieveCDCsForRegion("Northwest");
           mainAppletContext.fpNextItemPrompt.setDirection("right");
           mainAppletContext.fpNextItemPrompt.setLocation(lstStates.getLocation().x-20, lst
States.getLocation().y);
        }
        else if (blnMouseInNortheastArea){
           retrieveCDCsForRegion("Northeast");
           mainAppletContext.fpNextItemPrompt.setDirection("right");
           mainAppletContext.fpNextItemPrompt.setLocation(lstStates.getLocation().x-20, lst
States.getLocation().y);
        }
        else{
           lblSelectedRegion.setText("Selected Region: None");
        }
     }
     public void mouseReleased(MouseEvent me){}
     public void mouseDragged(MouseEvent me){}
     public void mouseMoved(MouseEvent me){
        mouseX = me.getX();
        mouseY = me.getY();
        if (plgSouthwestOutline.contains(mouseX, mouseY)){
```

```
            currentState = 1;
            blnMouseInSouthwestArea = true;
            blnMouseInSoutheastArea = false;
            blnMouseInNorthwestArea = false;
            blnMouseInNortheastArea = false;
            clipRectangle = plgSouthwestOutline.getBounds();
         }
         else if (plgSoutheastOutline.contains(mouseX, mouseY)){
            currentState = 2;
            blnMouseInSouthwestArea = false;
            blnMouseInSoutheastArea = true;
            blnMouseInNorthwestArea = false;
            blnMouseInNortheastArea = false;
            clipRectangle = plgSoutheastOutline.getBounds();
         }
         else if (plgNorthwestOutline.contains(mouseX, mouseY)){
            currentState = 3;
            blnMouseInSouthwestArea = false;
            blnMouseInSoutheastArea = false;
            blnMouseInNorthwestArea = true;
            blnMouseInNortheastArea = false;
            clipRectangle = plgNorthwestOutline.getBounds();
         }
         else if (plgNortheastOutline.contains(mouseX, mouseY)){
            currentState = 4;
            blnMouseInSouthwestArea = false;
            blnMouseInSoutheastArea = false;
            blnMouseInNorthwestArea = false;
            blnMouseInNortheastArea = true;
            clipRectangle = plgNortheastOutline.getBounds();
         }
         else{
            currentState = 5;
            blnMouseInSouthwestArea = false;
            blnMouseInSoutheastArea = false;
            blnMouseInNorthwestArea = false;
            blnMouseInNortheastArea = false;
            clipRectangle = new Rectangle(0, 0, getSize().width, getSize().height);
         }
         if (currentState != lastState){
            getGraphics().setClip(clipRectangle.getLocation().x, clipRectangle.getLocation()
.y, clipRectangle.getSize().width,
                                  clipRectangle.getSize().height);
            update(getGraphics());
            lastState = currentState;
            getGraphics().setClip(0, 0, getSize().width, getSize().height);
         }
      }
   }
}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
public class DartAppletInformationScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   Vector vtrSlides;
   int currentSlideIndex = 0;
   MediaTracker mt;
   SlideAdvanceOrDecrementCanvas slideAdvancer, slideDecrementer;
   Label lblPreviousSlide, lblNextSlide;
   PictureDisplayCanvas slideDisplayer;
   DartAppletChangeScreenButton backButton, forwardButton, homeButton;
   MouseOverButton btnReturnToDART, btnViewSurveyPlan, btnGISTutorial;
   RetrieveSlidesThread threadThatRetrievesTheSlides;
   boolean blnThreadThatRetrievesTheSlidesAlreadyStarted;

   public DartAppletInformationScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setLayout(null);

      //slides = new Image[11];
      vtrSlides = new Vector();

      lblNextSlide = new Label("Next");
      lblNextSlide.setSize(40, 20);
      lblNextSlide.setLocation(720, 190);
      add(lblNextSlide);

      slideAdvancer = new SlideAdvanceOrDecrementCanvas("advance");
      slideAdvancer.setSize(20, 80);
      slideAdvancer.setLocation(730, 220);
      add(slideAdvancer);

      lblPreviousSlide = new Label("Previous");
      lblPreviousSlide.setSize(50, 20);
      lblPreviousSlide.setLocation(10, 190);
      add(lblPreviousSlide);

      slideDecrementer = new SlideAdvanceOrDecrementCanvas("decrement");
      slideDecrementer.setSize(20, 80);
      slideDecrementer.setLocation(20, 220);
      add(slideDecrementer);

      slideDisplayer = new PictureDisplayCanvas();
      slideDisplayer.setSize(629, 472);
      slideDisplayer.setLocation(70, 60);
      add(slideDisplayer);

      btnReturnToDART = new MouseOverButton("Return To DART");
      btnReturnToDART.setSize(160, 20);
      btnReturnToDART.setLocation(120, 540);
      btnReturnToDART.addActionListener(this);
      btnReturnToDART.setFont(mainAppletContext.regularScreenFont);
      add(btnReturnToDART);

      btnViewSurveyPlan = new MouseOverButton("View Survey Plan");
      btnViewSurveyPlan.setSize(160, 20);
      btnViewSurveyPlan.setLocation(300, 540);
      btnViewSurveyPlan.addActionListener(this);
      btnViewSurveyPlan.setFont(mainAppletContext.regularScreenFont);
      add(btnViewSurveyPlan);
```

```
                if (threadIsPaused){
                   sleep(200);
                }
                else{
                   imgUnResizedImage = mainAppletContext.getImage(mainAppletContext.getDocume
ntBase(), "slides/slide" + (slideCount +1) + ".jpg");
                   mt.addImage(imgUnResizedImage, 0);
                   mt.waitForAll();
                   if (mt.isErrorAny()){
                      repaint();
                      break;
                   }

                   if ( (imgUnResizedImage.getWidth(null) == 627) && (imgUnResizedImage.getHe
ight(null) == 470) ){    // if already the

                        //right size don't rescale
                      vtrSlides.addElement( imgUnResizedImage );
                   }
                   else{   // otherwise make it the right size
                      vtrSlides.addElement( imgUnResizedImage.getScaledInstance(627, 470, Ima
ge.SCALE_SMOOTH) );
                      mt.addImage( (Image)(vtrSlides.elementAt(slideCount)), 0);
                      mt.waitForAll();
                      imgUnResizedImage.flush();
                   }

                   synchronized(htblRetrievedSlides){
                      htblRetrievedSlides.put( new Integer(slideCount), " ");
                   }
                   slideCount++;
                   repaint();
                }
             }
          }
          catch(Exception e){
             e.printStackTrace();
          }
       }
    }

    public class WaitOnSlideRetrievalThread extends Thread{
       int slideToWaitOn;
       public WaitOnSlideRetrievalThread(int slideToWaitOn){
          this.slideToWaitOn = slideToWaitOn;
       }
       public void run(){
          slideDisplayer.setWaitingForImage(true);
          while (! threadThatRetrievesTheSlides.isSlideRetrieved(slideToWaitOn) ){
             //System.out.println("waiting for slide");
             try{
                sleep(300);
             }
             catch (Exception e){
                e.printStackTrace();
             }
          }
          currentSlideIndex = slideToWaitOn;
          slideDisplayer.setImage((Image)(vtrSlides.elementAt(slideToWaitOn)));
          slideDisplayer.setWaitingForImage(false);
          repaint();
       }
    }

    public class PictureDisplayCanvas extends Canvas{
```

```
      Image pictureToDisplay;
      FontMetrics fm;
      String waitString = "Please wait, loading slide...";
      boolean blnWaitingForImageToLoad = false;
      public PictureDisplayCanvas(){
         setBackground(Color.white);
      }
      public void setImage(Image pictureToDisplay){
         this.pictureToDisplay = pictureToDisplay;
         repaint();
      }
      public void setWaitingForImage(boolean waiting){
        blnWaitingForImageToLoad = waiting;
        repaint();
      }
      public void paint(Graphics g){
         g.setFont(mainAppletContext.smallerScreenTitleFont);
         if (fm == null){
            fm = g.getFontMetrics();
         }
         g.setColor(Color.black);
         if (blnWaitingForImageToLoad){
            g.drawString(waitString, (int)(getSize().width/2.0 - fm.stringWidth(waitString)/
2.0 + 0.5), 100);
         }
         else{
            if (pictureToDisplay != null){
               g.drawImage(pictureToDisplay, 1, 1, null);
            }
         }
         g.drawRect(0, 0, getSize().width -1, getSize().height -1);
      }
      public void update(Graphics g){
         if (blnWaitingForImageToLoad){
            g.setColor(getBackground());
            g.fillRect(0, 0, getSize().width, getSize().height);
            paint(g);
         }
         else{
            paint(g);
         }
      }
   }

   public class SlideAdvanceOrDecrementCanvas extends Canvas implements MouseListener, Mouse
MotionListener{
      boolean blnMouseInside = false;
      boolean blnMouseInsidePreviously = false;
      Polygon plgSymbol;
      String direction;
      Image buffer;
      Graphics bufferG;
      public SlideAdvanceOrDecrementCanvas(String direction){
         this.direction = direction;
         addMouseListener(this);
         addMouseMotionListener(this);
      }
      public void paint(Graphics g){
         if (buffer == null){
            buffer = createImage(getSize().width, getSize().height);
            bufferG = buffer.getGraphics();
         }
         if (plgSymbol == null){
            plgSymbol = new Polygon();
            if (direction.equals("advance")){
```

```
            plgSymbol.addPoint(0, 0);
            plgSymbol.addPoint(0, getSize().height -1);
            plgSymbol.addPoint(getSize().width -1, (int)(getSize().height/2.0 + 0.5));
         }
         else if (direction.equals("decrement")){
            plgSymbol.addPoint(0, (int)(getSize().height/2.0 + 0.5));
            plgSymbol.addPoint(getSize().width -1, 0);
            plgSymbol.addPoint(getSize().width -1, getSize().height -1);
         }
      }
      if (blnMouseInside){
         bufferG.setColor(Color.blue);
      }
      else{
         bufferG.setColor(Color.black);
      }
      bufferG.fillPolygon(plgSymbol);
      g.drawImage(buffer, 0, 0, null);
   }
   public void update(Graphics g){
      paint(g);
   }
   public void mousePressed(MouseEvent me){}
   public void mouseReleased(MouseEvent me){}
   public void mouseClicked(MouseEvent me){
      if (blnMouseInside){
         if (direction.equals("advance")){
            displaySlide(currentSlideIndex + 1);
         }
         else if (direction.equals("decrement")){
            displaySlide(currentSlideIndex - 1);
         }
      }
   }
   public void mouseEntered(MouseEvent me){}
   public void mouseExited(MouseEvent me){
      blnMouseInside = false;
      repaint();
   }
   public void mouseMoved(MouseEvent me){
     blnMouseInsidePreviously = blnMouseInside;
     if (plgSymbol.contains(me.getX(), me.getY())){
        blnMouseInside = true;
     }
     else{
        blnMouseInside = false;
     }
     if ( blnMouseInsidePreviously != blnMouseInside){
        repaint();
     }
   }
   public void mouseDragged(MouseEvent me){}
}

}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.io.*;

public class DartAppletLEAInfoScreen extends Panel implements ActionListener, ItemListener,
KeyListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   MouseOverButton btnBack, btnNextPage, btnHome, btnCensusReport;
   MouseOverButton btnOverview;
   Label lblLEA, lblCurrentCDCoordinator, lblLEAName, lblJurisdictionType, lblJurisdictionNa
me, lblAddress,
         lblRankTitle, lblCity, lblContactState, lblZip, lblCommVoice, lblCommFax, lblDSNVoi
ce,
         lblDSNFax, lblEmail, lblURLWWWIntranet, lblUsername, lblPassword, lblLatitude, lblL
ongitude;
   List lstLEAs;
   TextArea txtAddress;
   TextField txtCurrentCDCoordinator, txtLEAName, txtJurisdictionName,
            txtRankTitle, txtCity, txtContactState, txtZip, txtCommVoice, txtCommFax, txtDS
NVoice, txtDSNFax,
            txtEmail, txtURLWWWIntranet, txtUsername, txtPassword, txtLatitude, txtLongitud
e;
   Vector vtrLEARecords;

   Image tempMapImage;
   Image mapOverviewImage;

   String strUsernameOfCDC;
   String strNameOfCDC;

   LEARecord currentRecord;

   Choice chcJurisdictionType;

   MouseOverButton btnEdit, btnCancel, btnSave;

   ConfirmEditWindow editConfirmationWindow;

   public DartAppletLEAInfoScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);

      lblLEA = new AutoLabel("LEAs:", mainAppletContext.regularScreenFont);
      lblLEA.setLocation(50, 120);
      add(lblLEA);

      lstLEAs = new List();
      lstLEAs.setSize(100, 120);
      lstLEAs.setLocation(50, 140);
      lstLEAs.addItemListener(this);
      add(lstLEAs);

      lblCurrentCDCoordinator = new AutoLabel("Current CD Coordinator:", mainAppletContext.r
egularScreenFont);
      lblCurrentCDCoordinator.setLocation(120, 120);
      //add(lblCurrentCDCoordinator);

      txtCurrentCDCoordinator = new TextField("");
      txtCurrentCDCoordinator.setSize(160, 20);
      txtCurrentCDCoordinator.setLocation(120, 140);
```

```
        //add(txtCurrentCDCoordinator);

        lblLEAName = new AutoLabel("LEA Name:", mainAppletContext.regularScreenFont);
        lblLEAName.setLocation(170, 120);
        add(lblLEAName);

        txtLEAName = new TextField("");
        txtLEAName.setSize(160, 20);
        txtLEAName.setLocation(170, 140);
        txtLEAName.addKeyListener(this);
        add(txtLEAName);

        lblJurisdictionType = new AutoLabel("Jurisdiction Type:", mainAppletContext.regularScr
eenFont);
        lblJurisdictionType.setLocation(170, 170);
        add(lblJurisdictionType);

        chcJurisdictionType = new Choice();
        chcJurisdictionType.setSize(160, 20);
        chcJurisdictionType.setLocation(170, 190);
        chcJurisdictionType.addItem("Select One");
        chcJurisdictionType.addItem("State");
        chcJurisdictionType.addItem("County");
        chcJurisdictionType.addItem("Urban");
        chcJurisdictionType.addItem("HIDTA");
        chcJurisdictionType.addItem("Special Agency Boundary");
        chcJurisdictionType.addItemListener(this);
        chcJurisdictionType.addKeyListener(this);
        add(chcJurisdictionType);

        lblJurisdictionName = new AutoLabel("Jurisdiction Name:", mainAppletContext.regularScr
eenFont);
        lblJurisdictionName.setLocation(170, 220);
        add(lblJurisdictionName);

        txtJurisdictionName = new TextField("");
        txtJurisdictionName.setSize(160, 20);
        txtJurisdictionName.setLocation(170, 240);
        txtJurisdictionName.addKeyListener(this);
        add(txtJurisdictionName);

        lblAddress = new AutoLabel("Address:", mainAppletContext.regularScreenFont);
        lblAddress.setLocation(370, 120);
        add(lblAddress);

        txtAddress = new TextArea("");
        txtAddress.setSize(260, 60);
        txtAddress.setLocation(370, 140);
        txtAddress.addKeyListener(this);
        add(txtAddress);

        lblCity = new AutoLabel("City:", mainAppletContext.regularScreenFont);
        lblCity.setLocation(50, 270);
        add(lblCity);

        txtCity = new TextField("");
        txtCity.setSize(100, 20);
        txtCity.setLocation(50, 290);
        txtCity.addKeyListener(this);
        add(txtCity);

        lblContactState = new AutoLabel("State:", mainAppletContext.regularScreenFont);
        lblContactState.setLocation(170, 270);
        add(lblContactState);
```

```
        txtContactState = new TextField("");
        txtContactState.setSize(100, 20);
        txtContactState.setLocation(170, 290);
        txtContactState.addKeyListener(this);
        add(txtContactState);

        lblZip = new AutoLabel("Zip:", mainAppletContext.regularScreenFont);
        lblZip.setLocation(50, 320);
        add(lblZip);

        txtZip = new TextField("");
        txtZip.setSize(100, 20);
        txtZip.setLocation(50, 340);
        txtZip.addKeyListener(this);
        add(txtZip);

        lblRankTitle = new AutoLabel("Rank/Title:", mainAppletContext.regularScreenFont);
        lblRankTitle.setLocation(170, 320);
        add(lblRankTitle);

        txtRankTitle = new TextField("");
        txtRankTitle.setSize(100, 20);
        txtRankTitle.setLocation(170, 340);
        txtRankTitle.addKeyListener(this);
        add(txtRankTitle);

        lblCommVoice = new AutoLabel("Comm. Voice:", mainAppletContext.regularScreenFont);
        lblCommVoice.setLocation(50, 370);
        add(lblCommVoice);

        txtCommVoice = new TextField("");
        txtCommVoice.setSize(100, 20);
        txtCommVoice.setLocation(50, 390);
        txtCommVoice.addKeyListener(this);
        add(txtCommVoice);

        lblCommFax = new AutoLabel("Comm. Fax:", mainAppletContext.regularScreenFont);
        lblCommFax.setLocation(170, 370);
        add(lblCommFax);

        txtCommFax = new TextField("");
        txtCommFax.setSize(100, 20);
        txtCommFax.setLocation(170, 390);
        txtCommFax.addKeyListener(this);
        add(txtCommFax);

        lblDSNVoice = new AutoLabel("DSN, Voice:", mainAppletContext.regularScreenFont);
        lblDSNVoice.setLocation(50, 420);
        add(lblDSNVoice);

        txtDSNVoice = new TextField("");
        txtDSNVoice.setSize(100, 20);
        txtDSNVoice.setLocation(50, 440);
        txtDSNVoice.addKeyListener(this);
        add(txtDSNVoice);

        lblDSNFax = new AutoLabel("DSN, Fax:", mainAppletContext.regularScreenFont);
        lblDSNFax.setLocation(170, 420);
        add(lblDSNFax);

        txtDSNFax = new TextField("");
        txtDSNFax.setSize(100, 20);
        txtDSNFax.setLocation(170, 440);
        txtDSNFax.addKeyListener(this);
        add(txtDSNFax);
```

```
        lblEmail = new AutoLabel("Email:", mainAppletContext.regularScreenFont);
        lblEmail.setLocation(50, 470);
        add(lblEmail);

        txtEmail = new TextField("");
        txtEmail.setSize(100, 20);
        txtEmail.setLocation(50, 490);
        txtEmail.addKeyListener(this);
        add(txtEmail);

        lblURLWWWIntranet = new AutoLabel("URL:", mainAppletContext.regularScreenFont);
        lblURLWWWIntranet.setLocation(170, 470);
        add(lblURLWWWIntranet);

        txtURLWWWIntranet = new TextField("");
        txtURLWWWIntranet.setSize(100, 20);
        txtURLWWWIntranet.setLocation(170, 490);
        txtURLWWWIntranet.addKeyListener(this);
        add(txtURLWWWIntranet);

        lblUsername = new AutoLabel("Username:", mainAppletContext.regularScreenFont);
        lblUsername.setLocation(50, 520);
        add(lblUsername);

        txtUsername = new TextField("");
        txtUsername.setSize(100, 20);
        txtUsername.setLocation(50, 540);
        txtUsername.addKeyListener(this);
        add(txtUsername);

        lblPassword = new AutoLabel("Password:", mainAppletContext.regularScreenFont);
        lblPassword.setLocation(170, 520);
        add(lblPassword);

        txtPassword = new TextField("");
        txtPassword.setSize(100, 20);
        txtPassword.setLocation(170, 540);
        txtPassword.addKeyListener(this);
        add(txtPassword);

        btnBack = new MouseOverButton("Back");
        btnBack.setSize(80, 20);
        btnBack.setLocation(320, 540);
        btnBack.addActionListener(this);
        btnBack.setFont(mainAppletContext.regularScreenFont);
        add(btnBack);

        btnNextPage = new MouseOverButton("Next Page");
        btnNextPage.setSize(80, 20);
        btnNextPage.setLocation(410, 540);
        btnNextPage.addActionListener(this);
        btnNextPage.setFont(mainAppletContext.regularScreenFont);
        add(btnNextPage);

        btnHome = new MouseOverButton("Home");
        btnHome.setSize(80, 20);
        btnHome.setLocation(500, 540);
        btnHome.addActionListener(this);
        btnHome.setFont(mainAppletContext.regularScreenFont);
        add(btnHome);

        btnCensusReport = new MouseOverButton("Census Report");
        btnCensusReport.setSize(110, 20);
        btnCensusReport.setLocation(590, 540);
```

```
        btnCensusReport.addActionListener(this);
        btnCensusReport.setFont(mainAppletContext.regularScreenFont);
        add(btnCensusReport);

        btnOverview = new MouseOverButton("Overview");
        btnOverview.setSize(70, 20);
        btnOverview.setLocation(694, 305);
        btnOverview.addActionListener(this);
        btnOverview.setFont(mainAppletContext.regularScreenFont);
        add(btnOverview);

        lblLatitude = new AutoLabel("Lat:", mainAppletContext.regularScreenFont);
        lblLatitude.setLocation(700, 400);
        add(lblLatitude);

        txtLatitude = new TextField("");
        txtLatitude.setSize(50, 20);
        txtLatitude.setLocation(700, 420);
        add(txtLatitude);

        lblLongitude = new AutoLabel("Lon:", mainAppletContext.regularScreenFont);
        lblLongitude.setLocation(700, 450);
        add(lblLongitude);

        txtLongitude = new TextField("");
        txtLongitude.setSize(50, 20);
        txtLongitude.setLocation(700, 470);
        add(txtLongitude);

        btnEdit = new MouseOverButton("Edit");
        btnEdit.setSize(60, 20);
        btnEdit.setLocation(694, 150);
        btnEdit.addActionListener(this);
        add(btnEdit);

        btnCancel = new MouseOverButton("Cancel");
        btnCancel.setSize(60, 20);
        btnCancel.setLocation(694, 150);
        btnCancel.addActionListener(this);
        btnCancel.setVisible(false);
        add(btnCancel);

        btnSave = new MouseOverButton("Save");
        btnSave.setSize(60, 20);
        btnSave.setLocation(694, 180);
        btnSave.addActionListener(this);
        btnSave.setVisible(false);
        add(btnSave);

        editConfirmationWindow = new ConfirmEditWindow();

    }

    public void setEnabledState(boolean enabledState){
        lstLEAs.setEnabled(enabledState);
        txtCurrentCDCoordinator.setEnabled(enabledState);
        txtLEAName.setEnabled(enabledState);
        chcJurisdictionType.setEnabled(enabledState);
        txtJurisdictionName.setEnabled(enabledState);
        txtAddress.setEnabled(enabledState);
        txtRankTitle.setEnabled(enabledState);
        txtCity.setEnabled(enabledState);
        txtContactState.setEnabled(enabledState);
        txtZip.setEnabled(enabledState);
        txtCommVoice.setEnabled(enabledState);
```

```
        txtCommFax.setEnabled(enabledState);
        txtDSNVoice.setEnabled(enabledState);
        txtDSNFax.setEnabled(enabledState);
        txtEmail.setEnabled(enabledState);
        txtURLWWWIntranet.setEnabled(enabledState);
        txtUsername.setEnabled(enabledState);
        txtPassword.setEnabled(enabledState);
        txtLatitude.setEnabled(enabledState);
        txtLongitude.setEnabled(enabledState);
    }

    public void setEditableState(boolean editableState){
        txtCurrentCDCoordinator.setEditable(editableState);
        txtLEAName.setEditable(editableState);
        chcJurisdictionType.setEnabled(editableState);
        txtJurisdictionName.setEditable(editableState);
        txtAddress.setEditable(editableState);
        txtRankTitle.setEditable(editableState);
        txtCity.setEditable(editableState);
        txtContactState.setEditable(editableState);
        txtZip.setEditable(editableState);
        txtCommVoice.setEditable(editableState);
        txtCommFax.setEditable(editableState);
        txtDSNVoice.setEditable(editableState);
        txtDSNFax.setEditable(editableState);
        txtEmail.setEditable(editableState);
        txtURLWWWIntranet.setEditable(editableState);
        txtUsername.setEditable(editableState);
        txtPassword.setEditable(editableState);
        txtLongitude.setEditable(editableState);
        txtLatitude.setEditable(editableState);

        txtCurrentCDCoordinator.setBackground(Color.white);     // setting to not editable has
the annoying effect of turning the text gray
        txtLEAName.setBackground(Color.white);
        chcJurisdictionType.setBackground(Color.white);
        txtJurisdictionName.setBackground(Color.white);
        txtAddress.setBackground(Color.white);
        txtRankTitle.setBackground(Color.white);
        txtCity.setBackground(Color.white);
        txtContactState.setBackground(Color.white);
        txtZip.setBackground(Color.white);
        txtCommVoice.setBackground(Color.white);
        txtCommFax.setBackground(Color.white);
        txtDSNVoice.setBackground(Color.white);
        txtDSNFax.setBackground(Color.white);
        txtEmail.setBackground(Color.white);
        txtURLWWWIntranet.setBackground(Color.white);
        txtUsername.setBackground(Color.white);
        txtPassword.setBackground(Color.white);
        txtLongitude.setBackground(Color.white);
        txtLatitude.setBackground(Color.white);
    }

    public void clearFields(){
        lstLEAs.removeAll();
        txtCurrentCDCoordinator.setText("");
        txtLEAName.setText("");
        chcJurisdictionType.select(0);
        txtJurisdictionName.setText("");
        txtAddress.setText("");
        txtRankTitle.setText("");
        txtCity.setText("");
        txtContactState.setText("");
        txtZip.setText("");
```

```
      txtCommVoice.setText("");
      txtCommFax.setText("");
      txtDSNVoice.setText("");
      txtDSNFax.setText("");
      txtEmail.setText("");
      txtURLWWWIntranet.setText("");
      txtUsername.setText("");
      txtPassword.setText("");
      txtLongitude.setText("");
      txtLatitude.setText("");
      /*btnEdit.setVisible(false);
      btnCancel.setVisible(false);
      btnSave.setVisible(false);*/
   }

   public void setCDC(CDCRecord currentCDC){
      strUsernameOfCDC = currentCDC.username;
      strNameOfCDC = currentCDC.name;
   }

   public void setCDC(String CDCUsername, String CDCName){
      strUsernameOfCDC = CDCUsername;
      strNameOfCDC = CDCName;
   }

   public void showRecord(int recordIndex){
      btnEdit.setVisible(false);

      currentRecord = (LEARecord)(vtrLEARecords.elementAt(recordIndex));
      setFields();

      if( (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.usern
ame.toUpperCase())) ||
          (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.CDCUs
ername.toUpperCase())) ){
         btnEdit.setVisible(true);
      }

      RetrieveLEAImageThread rlit = new RetrieveLEAImageThread( this);
      rlit.start();
   }

   class RetrieveLEAImageThread extends Thread{
      Component componentForMediaTracker;
      RetrieveLEAImageThread(Component componentForMediaTracker){
         this.componentForMediaTracker = componentForMediaTracker;
      }
      public void run(){
         try{
            if (tempMapImage != null){
               tempMapImage.flush();
               tempMapImage = null;
            }
            repaint();

            Object[] dataTransportArray = null;

            if (mainAppletContext.blnInNewLEAMode){    // if new user the picture will come
from the zip code lat/lon
               // do the zip -> lat/lon lookup here
               dataTransportArray = new Object[2];
               dataTransportArray[0] = "lookupLatAndLonForZip";
               dataTransportArray[1] = txtZip.getText();
               dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
               String[] latAndLonArray = (String[])dataTransportArray[0];
```

```
            currentRecord.lat = latAndLonArray[0];
            currentRecord.lon = latAndLonArray[1];
            txtLongitude.setText(currentRecord.lon);
            txtLatitude.setText(currentRecord.lat);

            dataTransportArray = new Object[8];
            dataTransportArray[0] = "UserImageRequest";
            dataTransportArray[1] = "LEAView";
            dataTransportArray[2] = txtLEAName.getText();
            dataTransportArray[3] = currentRecord.lat;    // these 2 will come from the d
atabase lookup
            dataTransportArray[4] = currentRecord.lon;
            dataTransportArray[5] = txtContactState.getText();
            dataTransportArray[6] = chcJurisdictionType.getSelectedItem();
            dataTransportArray[7] = txtJurisdictionName.getText();
            dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
         }
         else{         // the picture comes from their regular lat/lon
            dataTransportArray = new Object[8];
            dataTransportArray[0] = "UserImageRequest";
            dataTransportArray[1] = "LEAView";
            dataTransportArray[2] = currentRecord.name;
            dataTransportArray[3] = currentRecord.lat;
            dataTransportArray[4] = currentRecord.lon;
            dataTransportArray[5] = currentRecord.state;
            dataTransportArray[6] = currentRecord.jurisdictionType;
            dataTransportArray[7] = currentRecord.jurisdictionName;
            dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
         }

         String filePrefix = (String)dataTransportArray[0];
         String imageFileName = filePrefix + ".jpg";
         String fileFinishedMarkerFileName = filePrefix + ".finished";
         URL finishedFileURL = new URL(mainAppletContext.getCodeBase() + fileFinishedMark
erFileName);
         boolean fileCreated = false;
         int numberOfTries = 0;
         InputStream tempIS = null;
         while ( (! fileCreated) && (numberOfTries < 100) ){
            try{
               sleep(200);
               tempIS = finishedFileURL.openStream();
               tempIS.close();
               fileCreated = true;
            }
            catch(Exception fileNotYetThereException){
               numberOfTries++;
               if (numberOfTries == 100){
                  System.out.println("CDC image file not found");
               }
            }
         }
         if (fileCreated){
            Image unscaledTempMapImage = mainAppletContext.getImage(mainAppletContext.get
CodeBase(), imageFileName);
            MediaTracker mt = new MediaTracker(componentForMediaTracker);
            mt.addImage(unscaledTempMapImage, 0);
            mt.waitForAll();
            mt.removeImage(unscaledTempMapImage, 0);
            tempMapImage = unscaledTempMapImage.getScaledInstance( 342, 300, Image.SCALE_
SMOOTH);
            mt.addImage(tempMapImage, 0);
            mt.waitForAll();
            unscaledTempMapImage.flush();
```

```
                repaint();
                dataTransportArray = new Object[2];
                dataTransportArray[0] = "deleteImage";
                dataTransportArray[1] = imageFileName.substring(imageFileName.indexOf("/"), i
mageFileName.lastIndexOf("."));
                dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
            }
         }
         catch(Exception e){
            e.printStackTrace();
         }
      }
   }

   public void setFields(){
      txtCurrentCDCoordinator.setText(strNameOfCDC);
      txtLEAName.setText(currentRecord.name);
      chcJurisdictionType.select(currentRecord.jurisdictionType);
      txtJurisdictionName.setText(currentRecord.jurisdictionName);
      txtAddress.setText(currentRecord.address);
      txtRankTitle.setText(currentRecord.rank);
      txtCity.setText(currentRecord.city);
      txtContactState.setText(currentRecord.state);
      txtZip.setText(currentRecord.zip);
      txtCommVoice.setText(currentRecord.voice);
      txtCommFax.setText(currentRecord.fax);
      txtDSNVoice.setText(currentRecord.DSNVoice);
      txtDSNFax.setText(currentRecord.DSNFax);
      txtEmail.setText(currentRecord.email);
      txtURLWWWIntranet.setText(currentRecord.url);
      txtUsername.setText(currentRecord.username);
      txtPassword.setText(currentRecord.password);
      txtLongitude.setText(currentRecord.lon);
      txtLatitude.setText(currentRecord.lat);
   }

   public void setState1(){          // this state is where a LEA or CDC has logged in
      clearFields();
      setEnabledState(true);
      setEditableState(false);
      if (mainAppletContext.fpNextItemPrompt.getParent() == this){
         remove(mainAppletContext.fpNextItemPrompt);
      }

      lblLEA.setVisible(true);
      lstLEAs.setVisible(true);

      btnEdit.setVisible(false);
      btnSave.setVisible(false);
      btnCancel.setVisible(false);

      btnCensusReport.setEnabled(true);
      btnNextPage.setEnabled(true);
      btnBack.setEnabled(true);
      btnHome.setLabel("Home");
      btnHome.setEnabled(true);

      mainAppletContext.navigationPanel.enableCDCButton();
      mainAppletContext.navigationPanel.enableLEAButton();
      mainAppletContext.navigationPanel.enableHardwareButton();
      mainAppletContext.navigationPanel.enableOSSoftwareButton();
      mainAppletContext.navigationPanel.enableNetworkButton();
      mainAppletContext.navigationPanel.enableWebConnectButton();

      Object[] dataTransportArray = new Object[2];
```

```
      dataTransportArray[0] = "getLEAsForCDC";
      dataTransportArray[1] = strUsernameOfCDC;
      dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
      vtrLEARecords = (Vector)(dataTransportArray[0]);
      for (int i = 0; i < vtrLEARecords.size(); i++){
         lstLEAs.addItem( ((LEARecord)(vtrLEARecords.elementAt(i))).name);
      }
      if (lstLEAs.getItemCount() != 0){  // this just shows the logged in LEA if there is on
e. if not it shows the first record
         boolean blnUserFound = false;
         for (int i = 0; i < vtrLEARecords.size(); i++){
            if ( ( ((LEARecord)(vtrLEARecords.elementAt(i))).username.toUpperCase() ).equals
(
                                    mainAppletContext.strCurrentLoggedInUser.toUpperCase() ) ){
               blnUserFound = true;
               lstLEAs.select(i);
               lstLEAs.makeVisible(i);
               lstLEAsClicked();
               currentRecord = (LEARecord)vtrLEARecords.elementAt(i);
               lblUsername.setVisible(true);
               txtUsername.setVisible(true);
               lblPassword.setVisible(true);
               txtPassword.setVisible(true);
            }
         }
         if (! blnUserFound){
            lstLEAs.select(0);
            lstLEAsClicked();
            currentRecord = (LEARecord)vtrLEARecords.elementAt(0);
            lblUsername.setVisible(false);
            txtUsername.setVisible(false);
            lblPassword.setVisible(false);
            txtPassword.setVisible(false);
         }
         setFields();
      }
      else{   // avoid null pointers by not allowing them to view info of null LEAs
         btnNextPage.setEnabled(false);
         btnCensusReport.setEnabled(false);
         mainAppletContext.navigationPanel.disableLEAButton();
         mainAppletContext.navigationPanel.disableHardwareButton();
         mainAppletContext.navigationPanel.disableOSSoftwareButton();
         mainAppletContext.navigationPanel.disableNetworkButton();
         mainAppletContext.navigationPanel.disableWebConnectButton();
      }
   }

   public void setState2(){    // this state is where a new LEA is entering information and
has not gone through the 'script'
      clearFields();
      setEnabledState(false);
      setEditableState(true);
      if (mainAppletContext.fpNextItemPrompt.getParent() != null){
         mainAppletContext.fpNextItemPrompt.getParent().remove(mainAppletContext.fpNextItemP
rompt);
      }
      mainAppletContext.fpNextItemPrompt.setDirection("right");
      mainAppletContext.fpNextItemPrompt.setLocation(txtLEAName.getLocation().x - 20, txtLEA
Name.getLocation().y);
      add(mainAppletContext.fpNextItemPrompt);

      lblLEA.setVisible(false);
      lstLEAs.setVisible(false);

      btnEdit.setVisible(false);
```

```
      btnSave.setVisible(false);
      btnCancel.setVisible(false);

      btnCensusReport.setEnabled(false);
      btnNextPage.setEnabled(false);
      btnBack.setEnabled(true);
      btnHome.setLabel("Cancel");
      btnHome.setEnabled(true);

      mainAppletContext.navigationPanel.disableCDCButton();
      mainAppletContext.navigationPanel.disableLEAButton();
      mainAppletContext.navigationPanel.disableHardwareButton();
      mainAppletContext.navigationPanel.disableOSSoftwareButton();
      mainAppletContext.navigationPanel.disableNetworkButton();
      mainAppletContext.navigationPanel.disableWebConnectButton();

      currentRecord = mainAppletContext.lrNewLEARecord;

      lblUsername.setVisible(true);
      txtUsername.setVisible(true);
      lblPassword.setVisible(true);
      txtPassword.setVisible(true);

      setFields();
      txtLEAName.setEnabled(true);
      txtLEAName.setEditable(true);
      txtLEAName.requestFocus();

      mainAppletContext.wndNewLEAInformationWindow.setMessage("Please enter information abou
t yourself by filling " +
                                                              "in all of the following field
s.  If you do not know " +
                                                              "a value or it is not applicab
le, enter \"N/A\".  When " +
                                                              "you have finished, press \"Ce
nsus Report\" to continue.");
      mainAppletContext.wndNewLEAInformationWindow.show();
   }

   public void setState3(){      // this state is where a new LEA is entering information an
d has gone through the 'script'
      clearFields();
      setEnabledState(true);
      setEditableState(true);
      if (mainAppletContext.fpNextItemPrompt.getParent() != null){
         mainAppletContext.fpNextItemPrompt.getParent().remove(mainAppletContext.fpNextItemP
rompt);
      }

      lblLEA.setVisible(false);
      lstLEAs.setVisible(false);

      btnEdit.setVisible(false);
      btnSave.setVisible(false);
      btnCancel.setVisible(false);

      btnCensusReport.setEnabled(true);
      btnNextPage.setEnabled(false);
      btnBack.setEnabled(true);
      btnHome.setLabel("Cancel");
      btnHome.setEnabled(true);

      mainAppletContext.navigationPanel.disableCDCButton();
      mainAppletContext.navigationPanel.disableLEAButton();
      mainAppletContext.navigationPanel.disableHardwareButton();
```

```
        mainAppletContext.navigationPanel.disableOSSoftwareButton();
        mainAppletContext.navigationPanel.disableNetworkButton();
        mainAppletContext.navigationPanel.disableWebConnectButton();

        currentRecord = mainAppletContext.lrNewLEARecord;

        lblUsername.setVisible(true);
        txtUsername.setVisible(true);
        lblPassword.setVisible(true);
        txtPassword.setVisible(true);

        setFields();
    }

    public void setVisible(boolean visible){
        /*if (mainAppletContext.fpNextItemPrompt.getParent() == this){
            remove(mainAppletContext.fpNextItemPrompt);
        }*/
        if (visible){
            // clear maps
            if (tempMapImage != null){
                tempMapImage.flush();
                tempMapImage = null;
            }
            if (mapOverviewImage != null){
                mapOverviewImage.flush();
                mapOverviewImage = null;
            }

            if (! mainAppletContext.blnInNewLEAMode){
                setState1();
            }
            else if (! mainAppletContext.blnNewLEAHasGoneThroughScript){
                setState2();
            }
            else{
                setState3();
            }
            mainAppletContext.navigationPanel.pushLEAButton();
        }

        super.setVisible(visible);
    }

    public void setSelectedRecord(int recordIndex){
        currentRecord = (LEARecord)vtrLEARecords.elementAt(recordIndex);
    }

    public void disableNonEditActionComponents(){
        btnBack.setEnabled(false);
        btnNextPage.setEnabled(false);
        btnHome.setEnabled(false);
        btnCensusReport.setEnabled(false);
        btnOverview.setEnabled(false);
        lstLEAs.setEnabled(false);
        mainAppletContext.navigationPanel.disableCDCButton();
        mainAppletContext.navigationPanel.disableLEAButton();
        mainAppletContext.navigationPanel.disableHardwareButton();
        mainAppletContext.navigationPanel.disableOSSoftwareButton();
        mainAppletContext.navigationPanel.disableNetworkButton();
        mainAppletContext.navigationPanel.disableWebConnectButton();
    }

    public void enableNonEditActionComponents(){
        btnBack.setEnabled(true);
```

```
      btnNextPage.setEnabled(true);
      btnHome.setEnabled(true);
      btnCensusReport.setEnabled(true);
      btnOverview.setEnabled(true);
      lstLEAs.setEnabled(true);
      mainAppletContext.navigationPanel.enableCDCButton();
      mainAppletContext.navigationPanel.enableLEAButton();
      mainAppletContext.navigationPanel.enableHardwareButton();
      mainAppletContext.navigationPanel.enableOSSoftwareButton();
      mainAppletContext.navigationPanel.enableNetworkButton();
      mainAppletContext.navigationPanel.enableWebConnectButton();
   }

   public void btnBackClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrCDCInfoScreen);
   }

   public void btnNextPageClicked(){
      if (mainAppletContext.blnInNewLEAMode){
         currentRecord.username = txtUsername.getText();
         currentRecord.jurisdictionType = chcJurisdictionType.getSelectedItem();
         currentRecord.jurisdictionName = txtJurisdictionName.getText();
         currentRecord.rank = txtRankTitle.getText();
         currentRecord.name = txtLEAName.getText();
         currentRecord.CDCUsername = strUsernameOfCDC;
         currentRecord.address = txtAddress.getText();
         currentRecord.city = txtCity.getText();
         currentRecord.state = txtContactState.getText();
         currentRecord.zip = txtZip.getText();
         currentRecord.voice = txtCommVoice.getText();
         currentRecord.fax = txtCommFax.getText();
         currentRecord.DSNVoice = txtDSNVoice.getText();
         currentRecord.DSNFax = txtDSNFax.getText();
         currentRecord.email = txtEmail.getText();
         currentRecord.url = txtURLWWWIntranet.getText();
         currentRecord.password = txtPassword.getText();
         currentRecord.lon = txtLongitude.getText();
         currentRecord.lat = txtLatitude.getText();
      }
      mainAppletContext.showScreen(mainAppletContext.scrLEAHardwareScreen);
   }

   public void btnHomeClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrMainScreen);
   }

   public void btnCensusReportClicked(){
      if (mainAppletContext.blnInNewLEAMode){
         currentRecord.username = txtUsername.getText();
         currentRecord.jurisdictionType = chcJurisdictionType.getSelectedItem();
         currentRecord.jurisdictionName = txtJurisdictionName.getText();
         currentRecord.rank = txtRankTitle.getText();
         currentRecord.name = txtLEAName.getText();
         currentRecord.CDCUsername = strUsernameOfCDC;
         currentRecord.address = txtAddress.getText();
         currentRecord.city = txtCity.getText();
         currentRecord.state = txtContactState.getText();
         currentRecord.zip = txtZip.getText();
         currentRecord.voice = txtCommVoice.getText();
         currentRecord.fax = txtCommFax.getText();
         currentRecord.DSNVoice = txtDSNVoice.getText();
         currentRecord.DSNFax = txtDSNFax.getText();
         currentRecord.email = txtEmail.getText();
         currentRecord.url = txtURLWWWIntranet.getText();
         currentRecord.password = txtPassword.getText();
```

```
            currentRecord.lon = txtLongitude.getText();
            currentRecord.lat = txtLatitude.getText();
         }
         mainAppletContext.wndLEAITCensusWindow.show();
      }

      class RetrieveOverviewImageThread extends Thread{
         Component componentForMediaTracker;
         RetrieveOverviewImageThread(Component componentForMediaTracker){
            this.componentForMediaTracker = componentForMediaTracker;
         }
         public void run(){
            try{
               if (mapOverviewImage != null){
                  mapOverviewImage.flush();
                  mapOverviewImage = null;
               }
               repaint();

               Object[] dataTransportArray = new Object[8];
               dataTransportArray[0] = "UserImageRequest";
               dataTransportArray[1] = "LEAOVERVIEW";
               dataTransportArray[2] = currentRecord.name;
               dataTransportArray[3] = currentRecord.lat;
               dataTransportArray[4] = currentRecord.lon;
               dataTransportArray[5] = currentRecord.state;
               dataTransportArray[6] = currentRecord.jurisdictionType;
               dataTransportArray[7] = currentRecord.jurisdictionName;
               dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);

               String filePrefix = (String)dataTransportArray[0];
               String imageFileName = filePrefix + ".jpg";
               String fileFinishedMarkerFileName = filePrefix + ".finished";
               URL finishedFileURL = new URL(mainAppletContext.getCodeBase() + fileFinishedMark
erFileName);
               boolean fileCreated = false;
               int numberOfTries = 0;
               InputStream tempIS = null;
               while ( (! fileCreated) && (numberOfTries < 100) ){
                  try{
                     sleep(200);
                     tempIS = finishedFileURL.openStream();
                     tempIS.close();
                     fileCreated = true;
                  }
                  catch(Exception fileNotYetThereException){
                     numberOfTries++;
                     if (numberOfTries == 100){
                        System.out.println("LEA Overview image file not found");
                     }
                  }
               }
               if (fileCreated){
                  Image unscaledTempMapImage = mainAppletContext.getImage(mainAppletContext.get
CodeBase(), imageFileName);
                  MediaTracker mt = new MediaTracker(componentForMediaTracker);
                  mt.addImage(unscaledTempMapImage, 0);
                  mt.waitForAll();
                  mt.removeImage(unscaledTempMapImage, 0);
                  mapOverviewImage = unscaledTempMapImage.getScaledInstance( 160, 160, Image.SC
ALE_SMOOTH);
                  mt.addImage(mapOverviewImage, 0);
                  mt.waitForAll();
                  unscaledTempMapImage.flush();
                  //repaint();
```

```
                mainAppletContext.wndMapOverviewWindow.setImage(mapOverviewImage);
                mainAppletContext.wndMapOverviewWindow.show();
                dataTransportArray = new Object[2];
                dataTransportArray[0] = "deleteImage";
                dataTransportArray[1] = imageFileName.substring(imageFileName.indexOf("/"), i
mageFileName.lastIndexOf("."));
                dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
             }
          }
          catch(Exception e){
             e.printStackTrace();
          }
       }
    }

    public void btnOverviewClicked(){
       RetrieveOverviewImageThread roit = new RetrieveOverviewImageThread(this);
       roit.start();
    }

    public void btnEditClicked(){
       btnEdit.setVisible(false);
       btnCancel.setVisible(true);
       btnSave.setVisible(true);
       setEditableState(true);
       disableNonEditActionComponents();
    }

    public void btnCancelClicked(){
       btnCancel.setVisible(false);
       btnSave.setVisible(false);
       btnEdit.setVisible(true);
       setEditableState(false);
       setFields();   // this will set the textboxes back to the original value
       enableNonEditActionComponents();
    }

    public void btnSaveClicked(){
       editConfirmationWindow.show();
    }

    public void actionPerformed(ActionEvent ae){
       Object eventSource = ae.getSource();
       if (eventSource == btnBack){
          btnBackClicked();
       }
       else if (eventSource == btnNextPage){
          btnNextPageClicked();
       }
       else if (eventSource == btnHome){
          btnHomeClicked();
       }
       else if (eventSource == btnCensusReport){
          btnCensusReportClicked();
       }
       else if (eventSource == btnOverview){
          btnOverviewClicked();
       }
       else if (eventSource == btnEdit){
          btnEditClicked();
       }
       else if (eventSource == btnCancel){
          btnCancelClicked();
       }
       else if (eventSource == btnSave){
```

```
            btnSaveClicked();
         }
      }

      public void lstLEAsClicked(){
         if (mapOverviewImage != null){
            mapOverviewImage.flush();
            mapOverviewImage = null;
         }
         repaint();
         showRecord(lstLEAs.getSelectedIndex());
         btnNextPage.setEnabled(true);
         btnCensusReport.setEnabled(true);
         mainAppletContext.navigationPanel.enableHardwareButton();
         mainAppletContext.navigationPanel.enableOSSoftwareButton();
         mainAppletContext.navigationPanel.enableNetworkButton();
         mainAppletContext.navigationPanel.enableWebConnectButton();
      }

      public void itemStateChanged(ItemEvent ie){
         Object eventSource = ie.getSource();
         if (eventSource == lstLEAs){
            lstLEAsClicked();
         }
         else if (eventSource == chcJurisdictionType){
            if (mainAppletContext.blnInNewLEAMode){
               if (! chcJurisdictionType.getSelectedItem().equals("Select One")){
                  txtJurisdictionName.setEnabled(true);
                  chcJurisdictionType.transferFocus();
                  mainAppletContext.fpNextItemPrompt.setLocation(txtJurisdictionName.getLocatio
n().x - 20,
                                                                 txtJurisdictionName.getLocatio

n().y);
               }
            }
         }
      }

      public void paint(Graphics g){
         g.setFont(mainAppletContext.screenTitleFont);
         fm = g.getFontMetrics();

         tempString = "LEA Info for CDC " + strNameOfCDC;
         g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

         if (tempMapImage != null){
            g.drawImage(tempMapImage, 350, 230, 342, 300, this);
         }
         g.drawRect(349, 229, 343, 301);

         mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
      }

      public void updateRecordAndSendToServer(){
         currentRecord.jurisdictionType = chcJurisdictionType.getSelectedItem();
         currentRecord.jurisdictionName = txtJurisdictionName.getText();
         currentRecord.rank = txtRankTitle.getText();
         currentRecord.name = txtLEAName.getText();
         currentRecord.address = txtAddress.getText();
         currentRecord.city = txtCity.getText();
         currentRecord.state = txtContactState.getText();
         currentRecord.zip = txtZip.getText();
         currentRecord.voice = txtCommVoice.getText();
         currentRecord.fax = txtCommFax.getText();
```

```
        currentRecord.DSNVoice = txtDSNVoice.getText();
        currentRecord.DSNFax = txtDSNFax.getText();
        currentRecord.email = txtEmail.getText();
        currentRecord.url = txtURLWWWIntranet.getText();
        currentRecord.lon = txtLongitude.getText();
        currentRecord.lat = txtLatitude.getText();

        if (currentRecord.jurisdictionType.equals("")){
           currentRecord.jurisdictionType = " ";
        }
        if (currentRecord.jurisdictionName.equals("")){
           currentRecord.jurisdictionName = " ";
        }
        if (currentRecord.rank.equals("")){
           currentRecord.rank = " ";
        }
        if (currentRecord.name.equals("")){
           currentRecord.name = " ";
        }
        if (currentRecord.address.equals("")){
           currentRecord.address = " ";
        }
        if (currentRecord.city.equals("")){
           currentRecord.city = " ";
        }
        if (currentRecord.state.equals("")){
           currentRecord.state = " ";
        }
        if (currentRecord.zip.equals("")){
           currentRecord.zip = " ";
        }
        if (currentRecord.voice.equals("")){
           currentRecord.voice = " ";
        }
        if (currentRecord.fax.equals("")){
           currentRecord.fax = " ";
        }
        if (currentRecord.DSNVoice.equals("")){
           currentRecord.DSNVoice = " ";
        }
        if (currentRecord.DSNFax.equals("")){
           currentRecord.DSNFax = " ";
        }
        if (currentRecord.email.equals("")){
           currentRecord.email = " ";
        }
        if (currentRecord.url.equals("")){
           currentRecord.url = " ";
        }
        if (currentRecord.lon.equals("")){
           currentRecord.lon = "0.0";
        }
        if (currentRecord.lat.equals("")){
           currentRecord.lat = "0.0";
        }

        Object[] dataTransportArray = new Object[8];
        dataTransportArray[0] = "updateLEARecord";
        dataTransportArray[1] = currentRecord;
        dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
        enableNonEditActionComponents();
    }


    public void keyTyped(KeyEvent ke){}
```

```
   public void keyReleased(KeyEvent ke){}
   public void keyPressed(KeyEvent ke) {
      if ( (ke.getKeyCode() != KeyEvent.VK_ENTER) && (ke.getKeyCode() != KeyEvent.VK_TAB) ){
         return;
      }
      else{
         if (mainAppletContext.blnInNewLEAMode && !mainAppletContext.blnNewLEAHasGoneThrough
Script){
            Object eventSource = ke.getSource();
            ///((Component)(eventSource)).transferFocus();
            if (eventSource == txtLEAName){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(chcJurisdictionType.getLocatio
n().x - 20,
                                                              chcJurisdictionType.getLocatio
n().y);
               chcJurisdictionType.setEnabled(true);
            }
            else if (eventSource == chcJurisdictionType){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtJurisdictionName.getLocatio
n().x - 20,
                                                              txtJurisdictionName.getLocatio
n().y);
               txtJurisdictionName.setEnabled(true);
            }
            else if (eventSource == txtJurisdictionName){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtAddress.getLocation().x - 2
0, txtAddress.getLocation().y);
               txtAddress.setEnabled(true);
            }
            else if (eventSource == txtAddress){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtCity.getLocation().x - 20,
txtCity.getLocation().y);
               txtCity.setEnabled(true);
               ke.consume();
            }
            else if (eventSource == txtCity){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtContactState.getLocation().
x - 20, txtContactState.getLocation().y);
               txtContactState.setEnabled(true);
            }
            else if (eventSource == txtContactState){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtZip.getLocation().x - 20, t
xtZip.getLocation().y);
               txtZip.setEnabled(true);
            }
            else if (eventSource == txtZip){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtRankTitle.getLocation().x -
 20, txtRankTitle.getLocation().y);
               txtRankTitle.setEnabled(true);
               RetrieveLEAImageThread rlit = new RetrieveLEAImageThread( this);   // get the
 image for this zip
               rlit.start();
            }
            else if (eventSource == txtRankTitle){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtCommVoice.getLocation().x -
 20, txtCommVoice.getLocation().y);
               txtCommVoice.setEnabled(true);
```

```
            }
            else if (eventSource == txtCommVoice){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtCommFax.getLocation().x - 2
 txtCommFax.getLocation().y);
               txtCommFax.setEnabled(true);
            }
            else if (eventSource == txtCommFax){
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtDSNVoice.getLocation().x -
, txtDSNVoice.getLocation().y);
               txtDSNVoice.setEnabled(true);
            }
            else if (eventSource == txtDSNVoice){
               txtDSNFax.setEnabled(true);
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtDSNFax.getLocation().x - 20
 txtDSNFax.getLocation().y);
            }
            else if (eventSource == txtDSNFax){
               txtEmail.setEnabled(true);
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtEmail.getLocation().x - 20,
xtEmail.getLocation().y);
            }
            else if (eventSource == txtEmail){
               txtURLWWWIntranet.setEnabled(true);
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtURLWWWIntranet.getLocation(
x - 20, txtURLWWWIntranet.getLocation().y);
            }
            else if (eventSource == txtURLWWWIntranet){
               txtUsername.setEnabled(true);
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtUsername.getLocation().x -
 txtUsername.getLocation().y);
            }
            else if (eventSource == txtUsername){
               txtPassword.setEnabled(true);
               mainAppletContext.fpNextItemPrompt.setDirection("right");
               mainAppletContext.fpNextItemPrompt.setLocation(txtPassword.getLocation().x -
 txtPassword.getLocation().y);
            }
            else if (eventSource == txtPassword){
               btnCensusReport.setEnabled(true);
               mainAppletContext.fpNextItemPrompt.setDirection("down");
               mainAppletContext.fpNextItemPrompt.setLocation(btnCensusReport.getLocation().
 btnCensusReport.getLocation().y -20);
            }
            ((Component)(eventSource)).transferFocus();
         }
      }
   }


   class ConfirmEditWindow extends Frame implements ActionListener, WindowListener{
      MouseOverButton btnYes;
      MouseOverButton btnNo;
      AutoLabel lblConfirmMessage;
      ConfirmEditWindow(){
         setBackground(Color.lightGray);
         setResizable(false);
         setLayout(null);
         setSize(350, 170);
         setLocation(300, 300);
```

```
        setTitle("Save Changes Confirmation");

        lblConfirmMessage = new AutoLabel("Do you really want to save changes?", mainApplet
Context.regularScreenFont);
        lblConfirmMessage.setLocation(50, 60);
        add(lblConfirmMessage);

        btnYes = new MouseOverButton("Yes");
        btnYes.setSize(60, 20);
        btnYes.setLocation(100, 110);
        btnYes.addActionListener(this);
        add(btnYes);

        btnNo = new MouseOverButton("No");
        btnNo.setSize(60, 20);
        btnNo.setLocation(180, 110);
        btnNo.addActionListener(this);
        add(btnNo);

        addWindowListener(this);
    }
    public void btnYesClicked(){
        // send update to servlet
        updateRecordAndSendToServer();

        setEditableState(false);
        btnCancel.setVisible(false);
        btnSave.setVisible(false);
        btnEdit.setVisible(true);
        this.setVisible(false);
    }
    public void btnNoClicked(){
        this.setVisible(false);
    }
    public void actionPerformed(ActionEvent ae){
        Object eventSource = ae.getSource();
        if (eventSource == btnYes){
            btnYesClicked();
        }
        else if (eventSource == btnNo){
            btnNoClicked();
        }
    }

    public void windowActivated(WindowEvent we){}
    public void windowClosed(WindowEvent we){}
    public void windowClosing(WindowEvent we){
        this.setVisible(false);
    }
    public void windowDeactivated(WindowEvent we){
        this.setVisible(false);
    }
    public void windowIconified(WindowEvent we){}
    public void windowDeiconified(WindowEvent we){}
    public void windowOpened(WindowEvent we){}

}

}
```

```
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public final class DartServlet extends HttpServlet implements SingleThreadModel{
   ObjectInputStream objectReceiver = null;
   ObjectOutputStream objectSender = null;
   java.lang.Object transportObject = null;
   java.lang.Object[] dataTransportArray = null;
   String command = null;

   StringWriter exceptionStringWriter;
   PrintWriter exceptionPrintWriter;

   Connection dartDatabaseConnection;
   Statement currentStatement;
   ResultSet queryResultSet;

   static boolean blnIACLocked = false;

   public synchronized boolean lockIAC(){
      if (! blnIACLocked){
         blnIACLocked = true;
         return true;
      }
      return false;
   }

   public void unlockIAC(){
      blnIACLocked = false;
   }

   public void init(ServletConfig config) throws ServletException{
      super.init(config);
      exceptionStringWriter = new StringWriter();
      exceptionPrintWriter = new PrintWriter(exceptionStringWriter);

      try{
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         dartDatabaseConnection = DriverManager.getConnection("jdbc:odbc:DartDatabase", "",
"");
      }
      catch(Exception e){
         System.out.println("Exception in DartServlet init: " + e);
      }
   }

   public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletExceptio
n, IOException{
      PrintWriter out = res.getWriter();
      java.util.Date today = new java.util.Date();
      out.println(today.toString());
   }

   public void doPost(HttpServletRequest req, HttpServletResponse res){
      java.lang.Object transportObject = null;
      java.lang.Object returnObject = null;
      String command = null;

      String remoteHost = req.getRemoteHost();
      java.util.Date currentTime = new java.util.Date();
```

```
      try{
         objectReceiver = new ObjectInputStream(req.getInputStream());
         objectSender = new ObjectOutputStream(res.getOutputStream());
         transportObject = objectReceiver.readObject();
         dataTransportArray = (java.lang.Object[])transportObject;
         returnObject = null;
         command = (String)(dataTransportArray[0]);
      }
      catch(Exception e){
         getServletContext().log(e.getMessage(), e);
      }
      try{
         if (command.equals("getAllCDCData")){
            System.out.println("DART Servlet:  " + currentTime + " getAllCDCData command fro
m " + remoteHost);
            currentStatement = dartDatabaseConnection.createStatement();
            queryResultSet = currentStatement.executeQuery("Select * from [CDC DATA] order b
y state");
            Vector vtrCDCRecords = new Vector();
            CDCRecord currentRecord;
            while(queryResultSet.next()){
               currentRecord = new CDCRecord();
               currentRecord.username = queryResultSet.getString(1);
               currentRecord.password = queryResultSet.getString(2);
               currentRecord.name = queryResultSet.getString(3);
               currentRecord.service = queryResultSet.getString(4);
               currentRecord.rank = queryResultSet.getString(5);
               currentRecord.address = queryResultSet.getString(6);
               currentRecord.city = queryResultSet.getString(7);
               currentRecord.state = queryResultSet.getString(8);
               currentRecord.zip = queryResultSet.getString(9);
               currentRecord.lat = queryResultSet.getString(10);
               currentRecord.lon = queryResultSet.getString(11);
               currentRecord.voice = queryResultSet.getString(12);
               currentRecord.fax = queryResultSet.getString(13);
               currentRecord.DSNVoice = queryResultSet.getString(14);
               currentRecord.DSNFax = queryResultSet.getString(15);
               currentRecord.email = queryResultSet.getString(16);
               currentRecord.url = queryResultSet.getString(17);
               vtrCDCRecords.addElement(currentRecord);
            }
            returnObject = vtrCDCRecords;
            queryResultSet.close();
            currentStatement.close();
         }
         else if (command.equals("getSouthwestCDCData")){
            System.out.println("DART Servlet:  " + currentTime + " getSouthwestCDCData comma
nd from " + remoteHost);
            currentStatement = dartDatabaseConnection.createStatement();
            queryResultSet = currentStatement.executeQuery("Select * from [CDC DATA] where S
TATE in ('AZ', 'CA', 'CO', 'HI', 'NV', 'UT', 'NM', 'TX', 'OK', 'GU') order by state");
            Vector vtrCDCRecords = new Vector();
            CDCRecord currentRecord;
            while(queryResultSet.next()){
               currentRecord = new CDCRecord();
               currentRecord.username = queryResultSet.getString(1);
               currentRecord.password = queryResultSet.getString(2);
               currentRecord.name = queryResultSet.getString(3);
               currentRecord.service = queryResultSet.getString(4);
               currentRecord.rank = queryResultSet.getString(5);
               currentRecord.address = queryResultSet.getString(6);
               currentRecord.city = queryResultSet.getString(7);
               currentRecord.state = queryResultSet.getString(8);
               currentRecord.zip = queryResultSet.getString(9);
               currentRecord.lat = queryResultSet.getString(10);
```

```
                currentRecord.lon = queryResultSet.getString(11);
                currentRecord.voice = queryResultSet.getString(12);
                currentRecord.fax = queryResultSet.getString(13);
                currentRecord.DSNVoice = queryResultSet.getString(14);
                currentRecord.DSNFax = queryResultSet.getString(15);
                currentRecord.email = queryResultSet.getString(16);
                currentRecord.url = queryResultSet.getString(17);
                vtrCDCRecords.addElement(currentRecord);
             }
             returnObject = vtrCDCRecords;
             queryResultSet.close();
             currentStatement.close();
          }
          else if (command.equals("getSoutheastCDCData")){
             System.out.println("DART Servlet:  " + currentTime + " getSoutheastCDCData comma
nd from " + remoteHost);
             currentStatement = dartDatabaseConnection.createStatement();
             queryResultSet = currentStatement.executeQuery("Select * from [CDC DATA] where S
TATE in ('AL', 'AR', 'LA', 'MS', 'GA', 'FL', 'SC', 'NC', 'TN', 'KY', 'PR', 'VI') order by st
ate");
             Vector vtrCDCRecords = new Vector();
             CDCRecord currentRecord;
             while(queryResultSet.next()){
                currentRecord = new CDCRecord();
                currentRecord.username = queryResultSet.getString(1);
                currentRecord.password = queryResultSet.getString(2);
                currentRecord.name = queryResultSet.getString(3);
                currentRecord.service = queryResultSet.getString(4);
                currentRecord.rank = queryResultSet.getString(5);
                currentRecord.address = queryResultSet.getString(6);
                currentRecord.city = queryResultSet.getString(7);
                currentRecord.state = queryResultSet.getString(8);
                currentRecord.zip = queryResultSet.getString(9);
                currentRecord.lat = queryResultSet.getString(10);
                currentRecord.lon = queryResultSet.getString(11);
                currentRecord.voice = queryResultSet.getString(12);
                currentRecord.fax = queryResultSet.getString(13);
                currentRecord.DSNVoice = queryResultSet.getString(14);
                currentRecord.DSNFax = queryResultSet.getString(15);
                currentRecord.email = queryResultSet.getString(16);
                currentRecord.url = queryResultSet.getString(17);                vtrCDCRecords
.addElement(currentRecord);
             }
             returnObject = vtrCDCRecords;
             queryResultSet.close();
             currentStatement.close();
          }
          else if (command.equals("getNorthwestCDCData")){
             System.out.println("DART Servlet:  " + currentTime + " getNorthwestCDCData comma
nd from " + remoteHost);
             currentStatement = dartDatabaseConnection.createStatement();
             queryResultSet = currentStatement.executeQuery("Select * from [CDC DATA] where S
TATE in ('AK', 'WA', 'OR', 'ID', 'MT', 'WY', 'ND', 'SD', 'NE', 'KS', 'MN', 'IA', 'MO') order
 by state");
             Vector vtrCDCRecords = new Vector();
             CDCRecord currentRecord;
             while(queryResultSet.next()){
                currentRecord = new CDCRecord();
                currentRecord.username = queryResultSet.getString(1);
                currentRecord.password = queryResultSet.getString(2);
                currentRecord.name = queryResultSet.getString(3);
                currentRecord.service = queryResultSet.getString(4);
                currentRecord.rank = queryResultSet.getString(5);
                currentRecord.address = queryResultSet.getString(6);
                currentRecord.city = queryResultSet.getString(7);
```

```
                currentRecord.state = queryResultSet.getString(8);
                currentRecord.zip = queryResultSet.getString(9);
                currentRecord.lat = queryResultSet.getString(10);
                currentRecord.lon = queryResultSet.getString(11);
                currentRecord.voice = queryResultSet.getString(12);
                currentRecord.fax = queryResultSet.getString(13);
                currentRecord.DSNVoice = queryResultSet.getString(14);
                currentRecord.DSNFax = queryResultSet.getString(15);
                currentRecord.email = queryResultSet.getString(16);
                currentRecord.url = queryResultSet.getString(17);                vtrCDCRecords
.addElement(currentRecord);
             }
             returnObject = vtrCDCRecords;
             queryResultSet.close();
             currentStatement.close();
          }
          else if (command.equals("getNortheastCDCData")){
             System.out.println("DART Servlet:  " + currentTime + " getNortheastCDCData comma
nd from " + remoteHost);
             currentStatement = dartDatabaseConnection.createStatement();
             queryResultSet = currentStatement.executeQuery("Select * from [CDC DATA] where S
TATE in ('WI', 'IL', 'MI', 'IN', 'OH', 'WV', 'VA', 'PA', 'MD', 'DE', 'NJ', 'CT', 'RI', 'MA',
 'ME', 'NH', 'VT', 'NY', 'DC') order by state");
             Vector vtrCDCRecords = new Vector();
             CDCRecord currentRecord;
             while(queryResultSet.next()){
                currentRecord = new CDCRecord();
                currentRecord.username = queryResultSet.getString(1);
                currentRecord.password = queryResultSet.getString(2);
                currentRecord.name = queryResultSet.getString(3);
                currentRecord.service = queryResultSet.getString(4);
                currentRecord.rank = queryResultSet.getString(5);
                currentRecord.address = queryResultSet.getString(6);
                currentRecord.city = queryResultSet.getString(7);
                currentRecord.state = queryResultSet.getString(8);
                currentRecord.zip = queryResultSet.getString(9);
                currentRecord.lat = queryResultSet.getString(10);
                currentRecord.lon = queryResultSet.getString(11);
                currentRecord.voice = queryResultSet.getString(12);
                currentRecord.fax = queryResultSet.getString(13);
                currentRecord.DSNVoice = queryResultSet.getString(14);
                currentRecord.DSNFax = queryResultSet.getString(15);
                currentRecord.email = queryResultSet.getString(16);
                currentRecord.url = queryResultSet.getString(17);                vtrCDCRecords
.addElement(currentRecord);
             }
             returnObject = vtrCDCRecords;
             queryResultSet.close();
             currentStatement.close();
          }
          else if (command.equals("getLEAsForCDC")){
             System.out.println("DART Servlet:  " + currentTime + " getLEAsForCDC command fro
m " + remoteHost);
             String strCDC = (String)dataTransportArray[1];
             currentStatement = dartDatabaseConnection.createStatement();
             queryResultSet = currentStatement.executeQuery("Select * from [LEA DATA] where C
DC_User_Name = '" + strCDC + "' order by name");
             Vector vtrLEARecords = new Vector();
             LEARecord currentRecord;
             int columnIndex;
             while(queryResultSet.next()){
                currentRecord = new LEARecord();
                columnIndex = 1;
                currentRecord.username = queryResultSet.getString(columnIndex++);
                currentRecord.password = queryResultSet.getString(columnIndex++);
```

```
                currentRecord.CDCUsername = queryResultSet.getString(columnIndex++);
                currentRecord.name = queryResultSet.getString(columnIndex++);
                currentRecord.jurisdictionType = queryResultSet.getString(columnIndex++);
                currentRecord.jurisdictionName = queryResultSet.getString(columnIndex++);
                currentRecord.rank = queryResultSet.getString(columnIndex++);
                currentRecord.address = queryResultSet.getString(columnIndex++);
                currentRecord.city = queryResultSet.getString(columnIndex++);
                currentRecord.state = queryResultSet.getString(columnIndex++);
                currentRecord.zip = queryResultSet.getString(columnIndex++);
                currentRecord.lat = queryResultSet.getString(columnIndex++);
                currentRecord.lon = queryResultSet.getString(columnIndex++);
                currentRecord.voice = queryResultSet.getString(columnIndex++);
                currentRecord.fax = queryResultSet.getString(columnIndex++);
                currentRecord.DSNVoice = queryResultSet.getString(columnIndex++);
                currentRecord.DSNFax = queryResultSet.getString(columnIndex++);
                currentRecord.email = queryResultSet.getString(columnIndex++);
                currentRecord.url = queryResultSet.getString(columnIndex++);
                currentRecord.strAccessToComputer = queryResultSet.getString(columnIndex++);
                currentRecord.strTypeOfComputer = queryResultSet.getString(columnIndex++);
                currentRecord.strSpeedOfComputer = queryResultSet.getString(columnIndex++);
                currentRecord.strOperatingSystem = queryResultSet.getString(columnIndex++);
                currentRecord.strConnectedToNetwork = queryResultSet.getString(columnIndex++)
;
                currentRecord.strTypeOfNetworkConnection = queryResultSet.getString(columnInd
ex++);
                currentRecord.strSpeedOfNetworkConnection = queryResultSet.getString(columnIn
dex++);
                currentRecord.strBrowser = queryResultSet.getString(columnIndex++);
                currentRecord.strTypeOfBrowser = queryResultSet.getString(columnIndex++);
                currentRecord.strViewedWebPage = queryResultSet.getString(columnIndex++);
                vtrLEARecords.addElement(currentRecord);
            }
            returnObject = vtrLEARecords;
            queryResultSet.close();
            currentStatement.close();
        }
        else if (command.equals("updateCDCRecord")){
            System.out.println("DART Servlet:  " + currentTime + " updateCDCRecord command f
rom " + remoteHost);
            CDCRecord newCDCData = (CDCRecord)dataTransportArray[1];
            currentStatement = dartDatabaseConnection.createStatement();
            currentStatement.executeUpdate("Update [CDC Data] set  service = \'" +
                                           newCDCData.service + "\', rank = \'" +
                                           newCDCData.rank + "\', name = \'" + newCDCData.n
ame + "\', address = \'" +
                                           newCDCData.address + "\', city = \'" + newCDCDat
a.city + "\', state = \'" +
                                           newCDCData.state + "\', zip = \'" + newCDCData.z
ip + "\', lat = \'" +
                                           newCDCData.lat + "\', lon = \'" + newCDCData.lon
 + "\', comm_voice = \'" +
                                           newCDCData.voice + "\', comm_fax = \'" + newCDCD
ata.fax + "\', dsn_voice = \'" +
                                           newCDCData.DSNVoice + "\', dsn_fax = \'" + newCD
CData.DSNFax + "\', email = \'" +
                                           newCDCData.email + "\', url = \'" + newCDCData.u
rl + "\' where username = \'" +
                                           newCDCData.username + "\'");
            returnObject = "received";
            currentStatement.close();
        }
        else if (command.equals("updateLEARecord")){
            System.out.println("DART Servlet:  " + currentTime + " updateLEARecord command f
rom " + remoteHost);
            LEARecord newLEAData = (LEARecord)dataTransportArray[1];
```

```
        int hardwareValue1, hardwareValue2, hardwareValue3, softwareValue1, networkValue
, networkValue2,
            networkValue3, webconnectValue1, webconnectValue2, webconnectValue3;

        String temp;
        temp = newLEAData.strAccessToComputer;
        if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
           hardwareValue1 = 0;
        }
        else{
           hardwareValue1 = 1;
        }
        temp = newLEAData.strTypeOfComputer;
        if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
           hardwareValue2 = 0;
        }
        else{
           hardwareValue2 = 1;
        }
        temp = newLEAData.strSpeedOfComputer;
        if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
           hardwareValue3 = 0;
        }
        else{
           hardwareValue3 = 1;
        }
        temp = newLEAData.strOperatingSystem;
        if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
           softwareValue1 = 0;
        }
        else{
           softwareValue1 = 1;
        }
        temp = newLEAData.strConnectedToNetwork;
        if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
           networkValue1 = 0;
        }
        else{
           networkValue1 = 1;
        }
        temp = newLEAData.strTypeOfNetworkConnection;
        if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
           networkValue2 = 0;
        }
        else{
           networkValue2 = 1;
        }
        temp = newLEAData.strSpeedOfNetworkConnection;
        if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
           networkValue3 = 0;
        }
        else{
           networkValue3 = 1;
        }
        temp = newLEAData.strBrowser;
        if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
```

```
                webconnectValue1 = 0;
            }
            else{
                webconnectValue1 = 1;
            }
            temp = newLEAData.strTypeOfBrowser;
            if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
 || temp.equals("No") || temp.equals("None") ){
                webconnectValue2 = 0;
            }
            else{
                webconnectValue2 = 1;
            }
            temp = newLEAData.strViewedWebPage;
            if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
 || temp.equals("No") || temp.equals("None") ){
                webconnectValue3 = 0;
            }
            else{
                webconnectValue3 = 1;
            }

            int cummulativeValue = (hardwareValue1 + hardwareValue2 + hardwareValue3) * 1000
 + (softwareValue1) * 100 +
                                   (networkValue1 + networkValue2 + networkValue3) * 10 +
                                   (webconnectValue1 + webconnectValue2 + webconnectValue3);

            currentStatement = dartDatabaseConnection.createStatement();
            currentStatement.executeUpdate("Update [LEA Data] set  Jurisdiction_Type = \'" +
                                           newLEAData.jurisdictionType + "\', Jurisdiction_
Name = \'" +
                                           newLEAData.jurisdictionName + "\', rank = \'" +
newLEAData.rank + "\', name = \'" +
                                           newLEAData.name + "\', address = \'" + newLEADat
a.address + "\', city = \'" +
                                           newLEAData.city + "\', state = \'" + newLEAData.
state + "\', zip = \'" +
                                           newLEAData.zip + "\', comm_voice = \'" + newLEAD
ata.voice + "\', comm_fax = \'" +
                                           newLEAData.fax + "\', dsn_voice = \'" + newLEADa
ta.DSNVoice + "\', dsn_fax = \'" +
                                           newLEAData.DSNFax + "\', email = \'" + newLEADat
a.email + "\', url = \'" +
                                           newLEAData.url + "\', Access_to_computer = \'" +
                                           newLEAData.strAccessToComputer + "\', Type_of_co
mputer = \'" +
                                           newLEAData.strTypeOfComputer + "\', Speed_of_com
puter = \'" +
                                           newLEAData.strSpeedOfComputer + "\', Operating_s
ystem = \'" +
                                           newLEAData.strOperatingSystem + "\', Connected_t
o_network = \'" +
                                           newLEAData.strConnectedToNetwork + "\', Type_of_
network_connection = \'" +
                                           newLEAData.strTypeOfNetworkConnection + "\', Spe
ed_of_network_connection = \'" +
                                           newLEAData.strSpeedOfNetworkConnection + "\', Br
owser = \'" +
                                           newLEAData.strBrowser + "\', Name_of_browser = \
'" +
                                           newLEAData.strTypeOfBrowser + "\', Viewed_web_pa
ge = \'" +
                                           newLEAData.strViewedWebPage + "\', Access_to_com
puter_val = " +
                                           hardwareValue1 + ", Type_of_computer_val = " + h
```

```
ardwareValue2 + ", Speed_of_computer_val = " +
                                                hardwareValue3 + ", Operating_system_val = " + s
oftwareValue1 + ", Connected_to_network_val = " +
                                                networkValue1 + ", Type_of_network_connection_va
l = " + networkValue2 + ", Speed_of_network_connection_val = " +
                                                networkValue3 + ", Browser_val = " + webconnectV
alue1 + ", Name_of_browser_val = " + webconnectValue2 +
                                                ", Viewed_web_page_val = " + webconnectValue3 +
", Cummulative_val = " + cummulativeValue +
                                                " where username = \'" + newLEAData.username + "
\'");

            returnObject = "received";
            currentStatement.close();
         }
         else if (command.equals("newCDCRecord")){
         }
         else if (command.equals("newLEARecord")){
            System.out.println("DART Servlet:  " + currentTime + " newLEARecord command from
 " + remoteHost);
            LEARecord newRecord = (LEARecord)dataTransportArray[1];

            int hardwareValue1, hardwareValue2, hardwareValue3, softwareValue1, networkValue
1, networkValue2,
                networkValue3, webconnectValue1, webconnectValue2, webconnectValue3;

            String temp;
            temp = newRecord.strAccessToComputer;
            if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
 || temp.equals("No") || temp.equals("None") ){
               hardwareValue1 = 0;
            }
            else{
               hardwareValue1 = 1;
            }
            temp = newRecord.strTypeOfComputer;
            if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
 || temp.equals("No") || temp.equals("None") ){
               hardwareValue2 = 0;
            }
            else{
               hardwareValue2 = 1;
            }
            temp = newRecord.strSpeedOfComputer;
            if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
 || temp.equals("No") || temp.equals("None") ){
               hardwareValue3 = 0;
            }
            else{
               hardwareValue3 = 1;
            }
            temp = newRecord.strOperatingSystem;
            if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
 || temp.equals("No") || temp.equals("None") ){
               softwareValue1 = 0;
            }
            else{
               softwareValue1 = 1;
            }
            temp = newRecord.strConnectedToNetwork;
            if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
 || temp.equals("No") || temp.equals("None") ){
               networkValue1 = 0;
            }
            else{
```

```
            networkValue1 = 1;
         }
         temp = newRecord.strTypeOfNetworkConnection;
         if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
            networkValue2 = 0;
         }
         else{
            networkValue2 = 1;
         }
         temp = newRecord.strSpeedOfNetworkConnection;
         if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
            networkValue3 = 0;
         }
         else{
            networkValue3 = 1;
         }
         temp = newRecord.strBrowser;
         if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
            webconnectValue1 = 0;
         }
         else{
            webconnectValue1 = 1;
         }
         temp = newRecord.strTypeOfBrowser;
         if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
            webconnectValue2 = 0;
         }
         else{
            webconnectValue2 = 1;
         }
         temp = newRecord.strViewedWebPage;
         if ( temp.equals("Select One") || temp.equals("Unknown") || temp.equals("Other")
|| temp.equals("No") || temp.equals("None") ){
            webconnectValue3 = 0;
         }
         else{
            webconnectValue3 = 1;
         }

         int cummulativeValue = (hardwareValue1 + hardwareValue2 + hardwareValue3) * 1000
+ (softwareValue1) * 100 +
                                (networkValue1 + networkValue2 + networkValue3) * 10 +
                                (webconnectValue1 + webconnectValue2 + webconnectValue3);

         currentStatement = dartDatabaseConnection.createStatement();
         System.out.println("Insert into [LEA Data] values (" +
                                    "\'" + newRecord.username + "\'," +
                                    "\'" + newRecord.password + "\'," +
                                    "\'" + newRecord.CDCUsername + "\'," +
                                    "\'" + newRecord.name + "\'," +
                                    "\'" + newRecord.jurisdictionType + "\'," +
                                    "\'" + newRecord.jurisdictionName + "\'," +
                                    "\'" + newRecord.rank + "\'," +
                                    "\'" + newRecord.address + "\'," +
                                    "\'" + newRecord.city + "\'," +
                                    "\'" + newRecord.state + "\'," +
                                    "\'" + newRecord.zip + "\'," +
                                    "\'" + newRecord.lat + "\'," +
                                    "\'" + newRecord.lon + "\'," +
                                    "\'" + newRecord.voice + "\'," +
                                    "\'" + newRecord.fax + "\'," +
```

```
                                        "\'" + newRecord.DSNVoice + "\'," +
                                        "\'" + newRecord.DSNFax + "\'," +
                                        "\'" + newRecord.email + "\'," +
                                        "\'" + newRecord.url + "\'," +
                                        "\'" + newRecord.strAccessToComputer + "\'," +
                                        "\'" + newRecord.strTypeOfComputer + "\'," +
                                        "\'" + newRecord.strSpeedOfComputer + "\'," +
                                        "\'" + newRecord.strOperatingSystem + "\'," +
                                        "\'" + newRecord.strConnectedToNetwork + "\'," +
                                        "\'" + newRecord.strTypeOfNetworkConnection + "\'
,"  +
                                        "\'" + newRecord.strSpeedOfNetworkConnection + "\
',"  +
                                        "\'" + newRecord.strBrowser + "\'," +
                                        "\'" + newRecord.strTypeOfBrowser + "\'," +
                                        "\'" + newRecord.strViewedWebPage + "\'," +
                                       hardwareValue1 + "," + hardwareValue2 + "," + har
dwareValue3 + "," + softwareValue1 + "," +
                                       networkValue1 + "," + networkValue2 + "," + netwo
rkValue3 + "," + webconnectValue1 + "," +
                                       webconnectValue2 + "," + webconnectValue3 + "," +
 cummulativeValue +
                                       ")");
            currentStatement.executeUpdate("Insert into [LEA Data] values (" +
                                        "\'" + newRecord.username + "\'," +
                                        "\'" + newRecord.password + "\'," +
                                        "\'" + newRecord.CDCUsername + "\'," +
                                        "\'" + newRecord.name + "\'," +
                                        "\'" + newRecord.jurisdictionType + "\'," +
                                        "\'" + newRecord.jurisdictionName + "\'," +
                                        "\'" + newRecord.rank + "\'," +
                                        "\'" + newRecord.address + "\'," +
                                        "\'" + newRecord.city + "\'," +
                                        "\'" + newRecord.state + "\'," +
                                        "\'" + newRecord.zip + "\'," +
                                        "\'" + newRecord.lat + "\'," +
                                        "\'" + newRecord.lon + "\'," +
                                        "\'" + newRecord.voice + "\'," +
                                        "\'" + newRecord.fax + "\'," +
                                        "\'" + newRecord.DSNVoice + "\'," +
                                        "\'" + newRecord.DSNFax + "\'," +
                                        "\'" + newRecord.email + "\'," +
                                        "\'" + newRecord.url + "\'," +
                                        "\'" + newRecord.strAccessToComputer + "\'," +
                                        "\'" + newRecord.strTypeOfComputer + "\'," +
                                        "\'" + newRecord.strSpeedOfComputer + "\'," +
                                        "\'" + newRecord.strOperatingSystem + "\'," +
                                        "\'" + newRecord.strConnectedToNetwork + "\'," +
                                        "\'" + newRecord.strTypeOfNetworkConnection + "\'
,"  +
                                        "\'" + newRecord.strSpeedOfNetworkConnection + "\
',"  +
                                        "\'" + newRecord.strBrowser + "\'," +
                                        "\'" + newRecord.strTypeOfBrowser + "\'," +
                                        "\'" + newRecord.strViewedWebPage + "\'," +
                                       hardwareValue1 + "," + hardwareValue2 + "," + har
dwareValue3 + "," + softwareValue1 + "," +
                                       networkValue1 + "," + networkValue2 + "," + netwo
rkValue3 + "," + webconnectValue1 + "," +
                                       webconnectValue2 + "," + webconnectValue3 + "," +
 cummulativeValue +
                                       ")");
            returnObject = "received";
            currentStatement.close();
        }
```

```
        else if (command.equals("UserImageRequest")){

          String strViewType = (String)dataTransportArray[1];   // draw only current LEA, n
ot sure what this does on a cdc view...
          String strName = (String)dataTransportArray[2];
          String strLat = (String)dataTransportArray[3];
          String strLon = (String)dataTransportArray[4];
          String strState = (String)dataTransportArray[5];
          String strJurisdictionType = (String)dataTransportArray[6];
          String strJurisdictionName = (String)dataTransportArray[7];

          String strFilename = remoteHost + "_" + System.currentTimeMillis();

          String strFilenameWithPath = "c:\\inetpub\\wwwroot\\DartApplet\\ImageRequests\\"
+ strFilename;
          String strFilenameForClient = "ImageRequests/" + strFilename;

          String[] commandArray = new String[10];
          commandArray[0] = "d:\\dartarcview\\iac.exe";
          commandArray[1] = "127.0.0.1";
          commandArray[2] = strViewType;
          commandArray[3] = "\"" + strName + "\"";
          commandArray[4] = strLat;
          commandArray[5] = strLon;
          commandArray[6] = strState;
          commandArray[7] = "\"" + strJurisdictionType + "\"";
          commandArray[8] = "\"" + strJurisdictionName + "\"";
          commandArray[9] = strFilenameWithPath;

          while (! lockIAC()){  // get the lock for IAC.exe  (we only want it executing one
 at the time  (if iac.exe is called more than once at the same time, arcview crashes)
              Thread.currentThread().sleep(100);
          }

          Runtime.getRuntime().exec(commandArray);

          /*Process p = Runtime.getRuntime().exec(commandArray);
          InputStream is = p.getInputStream();
          System.out.print("Output from iac.exe call: ");
          int nextByte = 0;
          while( (nextByte = is.read()) != -1 ){
             System.out.print( (char)nextByte );
          }
          System.out.println();*/

          unlockIAC();     // let some other servlet call IAC.exe

          returnObject = strFilenameForClient;
        }
        else if (command.equals("GISTutorialImageRequest")){
          String strBitString = (String)dataTransportArray[1];
          String strFilename = remoteHost + "_" + System.currentTimeMillis();

          String strFilenameWithPath = "c:\\inetpub\\wwwroot\\DartApplet\\ImageRequests\\"
+ strFilename;
          String strFilenameForClient = "ImageRequests/" + strFilename;

          String[] commandArray = new String[5];
          commandArray[0] = "d:\\dartarcview\\iac.exe";
          commandArray[1] = "127.0.0.1";
          commandArray[2] = "TutorialView";
          commandArray[3] = strBitString;
          commandArray[4] = strFilenameWithPath;

          while (! lockIAC()){  // get the lock for IAC.exe  (we only want it executing one
```

```
at the time
            Thread.currentThread().sleep(100);
         }

         Runtime.getRuntime().exec(commandArray);
         /*Process p = Runtime.getRuntime().exec(commandArray);
         InputStream is = p.getInputStream();
         System.out.print("Output from iac.exe call: ");
         int nextByte = 0;
         while( (nextByte = is.read()) != -1 ){
            System.out.print( (char)nextByte );
         }
         System.out.println();*/

         unlockIAC();

         returnObject = strFilenameForClient;
      }
      else if (command.equals("deleteImage")){
         String strFileToDelete = "c:\\inetpub\\wwwroot\\DartApplet\\ImageRequests\\" + (
String)dataTransportArray[1];
         File fileToDelete = new File(strFileToDelete + ".jpg");
         fileToDelete.delete();
         fileToDelete = new File(strFileToDelete + ".finished");
         fileToDelete.delete();
         returnObject = "received";
      }
      else if (command.equals("lookupLatAndLonForZip")){
         String zip = (String)dataTransportArray[1];
         String lat = null;
         String lon = null;
         currentStatement = dartDatabaseConnection.createStatement();
         queryResultSet = currentStatement.executeQuery("Select latitude, longitude from
Zips where zipCode = '" + zip + "'");
         boolean blnRowFound = queryResultSet.next();
         if (blnRowFound){
            lat = queryResultSet.getString(1);
            lon = queryResultSet.getString(2);
         }
         String[] latAndLon = {lat, lon};
         returnObject = latAndLon;
      }
      else if (command.equals("checkUsernameAndPassword")){
         currentStatement = dartDatabaseConnection.createStatement();
         if ( ((String)dataTransportArray[1]).equals("CDC")){
            queryResultSet = currentStatement.executeQuery("Select password, state, Usern
ame, Name from [CDC Data] where username = '" + (String)(dataTransportArray[2]) + "'");
         }
         else if ( ((String)dataTransportArray[1]).equals("LEA")){
            queryResultSet = currentStatement.executeQuery("Select password, state, CDC_U
ser_Name from [LEA Data] where username = '" + (String)(dataTransportArray[2]) + "'");
         }

         String strAnswer;
         String strState;
         String strCDCUsername;
         String strCDCName = null;
         boolean blnRowFound = queryResultSet.next();

         if (blnRowFound && ( (queryResultSet.getString(1).toUpperCase()).equals( ((Strin
g)(dataTransportArray[3])).toUpperCase() ))){
            System.out.println("DART Servlet:  " + currentTime + " checkUsernameAndPasswo
rd command from " + remoteHost + ", user/pass accepted");
            strAnswer = "accepted";
            strState = queryResultSet.getString(2).toUpperCase();
```

```
                strCDCUsername = queryResultSet.getString(3).toUpperCase();
                if ( ((String)dataTransportArray[1]).equals("CDC")){
                   strCDCName = queryResultSet.getString(4);
                }
                else{
                   queryResultSet.close();
                   currentStatement.close();
                   currentStatement = dartDatabaseConnection.createStatement();
                   queryResultSet = currentStatement.executeQuery("Select Name from [CDC Data
] where Username = '" + strCDCUsername +  "'");
                   queryResultSet.next();
                   strCDCName = queryResultSet.getString(1);
                }
             }
             else{
                System.out.println("DART Servlet:  " + currentTime + " checkUsernameAndPasswo
rd command from " + remoteHost + ", user/pass rejected");
                strAnswer = "rejected";
                strState = "no state";
                strCDCUsername = "none";
             }
             String[] results = new String[4];
             results[0] = strAnswer;
             results[1] = strState;
             results[2] = strCDCUsername;
             results[3] = strCDCName;
             returnObject = results;
             queryResultSet.close();
             currentStatement.close();
          }
          else{
             System.out.println("Unrecognized command sent by client");
          }
          dataTransportArray = new java.lang.Object[1];
          dataTransportArray[0] = returnObject;
       }
       catch(Exception e){
          System.out.println("" + e + " exception on command " + command + " from " + req.get
RemoteHost());
          exceptionPrintWriter.write("Exception on command: " + command + "\n");
          e.printStackTrace(exceptionPrintWriter);
          getServletContext().log(e.getMessage(), e);
          dataTransportArray = new java.lang.Object[2];
          dataTransportArray[0] = returnObject;
          String header = "\nServlet threw an Exception: \n";
          exceptionPrintWriter.flush();
          dataTransportArray[1] = header + exceptionStringWriter.toString();
          exceptionStringWriter.getBuffer().setLength(0);
       }
       try{
          objectSender.writeObject(dataTransportArray);
          objectSender.flush();
       }
       catch(Exception anotherE){
          getServletContext().log(anotherE.getMessage(), anotherE);
       }

    }
}
```

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;

public class DartAppletNavigationPanel extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   NavigationButton nbCDC, nbLEA, nbHardware, nbOSSoftware, nbNetwork, nbWebConnect;

   NavigationButton currentlySelectedButton;

   PopupMenu pmCDC, pmLEA, pmHardware, pmOSSoftware, pmNetwork, pmWebConnect;

   DartAppletNavigationPanel(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);
      setSize(700, 20);
      setLocation(0, 0);

      nbCDC = new NavigationButton("CDC Info", mainAppletContext.regularScreenFont);
      nbCDC.setSize(100, 20);
      nbCDC.setLocation(0, 0);
      add(nbCDC);

      nbLEA = new NavigationButton("LEA Info", mainAppletContext.regularScreenFont);
      nbLEA.setSize(100, 20);
      nbLEA.setLocation(120, 0);
      add(nbLEA);

      nbHardware = new NavigationButton("Hardware", mainAppletContext.regularScreenFont);
      nbHardware.setSize(100, 20);
      nbHardware.setLocation(240, 0);
      add(nbHardware);

      nbOSSoftware = new NavigationButton("OS/Software", mainAppletContext.regularScreenFont
);
      nbOSSoftware.setSize(100, 20);
      nbOSSoftware.setLocation(360, 0);
      add(nbOSSoftware);

      nbNetwork = new NavigationButton("Network", mainAppletContext.regularScreenFont);
      nbNetwork.setSize(100, 20);
      nbNetwork.setLocation(480, 0);
      add(nbNetwork);

      nbWebConnect = new NavigationButton("Web Connect", mainAppletContext.regularScreenFont
);
      nbWebConnect.setSize(100, 20);
      nbWebConnect.setLocation(600, 0);
      add(nbWebConnect);

      pmCDC = new PopupMenu();
      MenuItem miCDC1 = new MenuItem("View Info");
      miCDC1.setFont(mainAppletContext.regularScreenFont);
      miCDC1.setActionCommand("ViewCDCInfo");
      miCDC1.addActionListener(this);
      MenuItem miCDC2 = new MenuItem("Edit Info");
      miCDC2.setFont(mainAppletContext.regularScreenFont);
      miCDC2.setActionCommand("EditCDCInfo");
      miCDC2.addActionListener(this);
      MenuItem miCDC3 = new MenuItem("Email CDC");
      miCDC3.setFont(mainAppletContext.regularScreenFont);
      miCDC3.setActionCommand("EmailCDC");
      miCDC3.addActionListener(this);
      pmCDC.add(miCDC1);
```

```
        pmCDC.add(miCDC2);
        pmCDC.add(miCDC3);
        add(pmCDC);

        pmLEA = new PopupMenu();
        MenuItem miLEA1 = new MenuItem("View Info");
        miLEA1.setFont(mainAppletContext.regularScreenFont);
        miLEA1.setActionCommand("ViewLEAInfo");
        miLEA1.addActionListener(this);
        MenuItem miLEA2 = new MenuItem("Edit Info");
        miLEA2.setFont(mainAppletContext.regularScreenFont);
        miLEA2.setActionCommand("EditLEAInfo");
        miLEA2.addActionListener(this);
        MenuItem miLEA3 = new MenuItem("Add New LEA");
        miLEA3.setFont(mainAppletContext.regularScreenFont);
        miLEA3.setActionCommand("AddNewLEA");
        miLEA3.addActionListener(this);
        MenuItem miLEA4 = new MenuItem("Statistics");
        miLEA4.setFont(mainAppletContext.regularScreenFont);
        miLEA4.setActionCommand("LEAStatistics");
        miLEA4.addActionListener(this);
        MenuItem miLEA5 = new MenuItem("Email LEA");
        miLEA5.setFont(mainAppletContext.regularScreenFont);
        miLEA5.setActionCommand("EmailLEA");
        miLEA5.addActionListener(this);
        pmLEA.add(miLEA1);
        pmLEA.add(miLEA2);
        pmLEA.add(miLEA3);
        pmLEA.add(miLEA4);
        pmLEA.add(miLEA5);
        add(pmLEA);

        pmHardware = new PopupMenu();
        MenuItem miHardware1 = new MenuItem("View Info");
        miHardware1.setFont(mainAppletContext.regularScreenFont);
        miHardware1.setActionCommand("ViewHardwareInfo");
        miHardware1.addActionListener(this);
        MenuItem miHardware2 = new MenuItem("Edit Info");
        miHardware2.setFont(mainAppletContext.regularScreenFont);
        miHardware2.setActionCommand("EditHardwareInfo");
        miHardware2.addActionListener(this);
        pmHardware.add(miHardware1);
        pmHardware.add(miHardware2);
        add(pmHardware);

        pmOSSoftware = new PopupMenu();
        MenuItem miOSSoftware1 = new MenuItem("View Info");
        miOSSoftware1.setFont(mainAppletContext.regularScreenFont);
        miOSSoftware1.setActionCommand("ViewOSSoftwareInfo");
        miOSSoftware1.addActionListener(this);
        MenuItem miOSSoftware2 = new MenuItem("Edit Info");
        miOSSoftware2.setFont(mainAppletContext.regularScreenFont);
        miOSSoftware2.setActionCommand("EditOSSoftwareInfo");
        miOSSoftware2.addActionListener(this);
        pmOSSoftware.add(miOSSoftware1);
        pmOSSoftware.add(miOSSoftware2);
        add(pmOSSoftware);

        pmNetwork = new PopupMenu();
        MenuItem miNetwork1 = new MenuItem("View Info");
        miNetwork1.setFont(mainAppletContext.regularScreenFont);
        miNetwork1.setActionCommand("ViewNetworkInfo");
        miNetwork1.addActionListener(this);
        MenuItem miNetwork2 = new MenuItem("Edit Info");
        miNetwork2.setFont(mainAppletContext.regularScreenFont);
```

```
      miNetwork2.setActionCommand("EditNetworkInfo");
      miNetwork2.addActionListener(this);
      pmNetwork.add(miNetwork1);
      pmNetwork.add(miNetwork2);
      add(pmNetwork);

      pmWebConnect = new PopupMenu();
      MenuItem miWebConnect1 = new MenuItem("View Info");
      miWebConnect1.setFont(mainAppletContext.regularScreenFont);
      miWebConnect1.setActionCommand("ViewWebConnectInfo");
      miWebConnect1.addActionListener(this);
      MenuItem miWebConnect2 = new MenuItem("Edit Info");
      miWebConnect2.setFont(mainAppletContext.regularScreenFont);
      miWebConnect2.setActionCommand("EditWebConnectInfo");
      miWebConnect2.addActionListener(this);
      pmWebConnect.add(miWebConnect1);
      pmWebConnect.add(miWebConnect2);
      add(pmWebConnect);
   }

   public void viewCDCInfoClicked(){
      if (! mainAppletContext.scrITCensusScreen1.txtCDCName.getText().equals("")){
         mainAppletContext.scrCDCInfoScreen.setCDCRecords(mainAppletContext.scrITCensusScree
n1.vtrCDCRecords);
         mainAppletContext.scrCDCInfoScreen.setSelectedRecord(mainAppletContext.scrITCensusS
creen1.currentlySelectedIndex);
         mainAppletContext.showScreen(mainAppletContext.scrCDCInfoScreen);
      }
   }
   public void editCDCInfoClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrCDCInfoScreen);
      if (mainAppletContext.scrCDCInfoScreen.btnEdit.isVisible()){
         mainAppletContext.scrCDCInfoScreen.btnEditClicked();
      }
   }
   public void emailCDCClicked(){
      try{
         mainAppletContext.getAppletContext().showDocument(new URL("mailto:" + ( ((CDCRecord
)(mainAppletContext.scrITCensusScreen1.vtrCDCRecords.elementAt(mainAppletContext.scrITCensus
Screen1.lstStates.getSelectedIndex()))).email )));
      }
      catch(Exception e){
         e.printStackTrace();
      }
   }
   public void viewLEAInfoClicked(){
      mainAppletContext.scrLEAInfoScreen.setCDC( mainAppletContext.scrCDCInfoScreen.currentR
ecord );
      mainAppletContext.showScreen(mainAppletContext.scrLEAInfoScreen);
   }
   public void editLEAInfoClicked(){
      if (mainAppletContext.scrCDCInfoScreen.isVisible()){  // only set the cdc if it hasnt
already been set (and thats when you are going from cdc screen to lea screen)
         mainAppletContext.scrLEAInfoScreen.setCDC( mainAppletContext.scrCDCInfoScreen.curre
ntRecord );
      }
      if (! mainAppletContext.scrLEAInfoScreen.isVisible()){
         mainAppletContext.showScreen(mainAppletContext.scrLEAInfoScreen);
      }
      if (mainAppletContext.scrLEAInfoScreen.btnEdit.isVisible()){
         mainAppletContext.scrLEAInfoScreen.btnEditClicked();
      }
   }
   public void addNewLEAClicked(){
      mainAppletContext.blnInNewLEAMode = true;
```

```
      mainAppletContext.blnNewLEAHasGoneThroughScript = false;
      mainAppletContext.lrNewLEARecord = new LEARecord();

      mainAppletContext.lrNewLEARecord.username = "";
      mainAppletContext.lrNewLEARecord.jurisdictionType = "Select One";
      mainAppletContext.lrNewLEARecord.jurisdictionName = "";
      mainAppletContext.lrNewLEARecord.rank = "";
      mainAppletContext.lrNewLEARecord.name = "";
      mainAppletContext.lrNewLEARecord.CDCUsername = "";
      mainAppletContext.lrNewLEARecord.address = "";
      mainAppletContext.lrNewLEARecord.city = "";
      mainAppletContext.lrNewLEARecord.state = "";
      mainAppletContext.lrNewLEARecord.zip = "";
      mainAppletContext.lrNewLEARecord.lat = "0.0";
      mainAppletContext.lrNewLEARecord.lon = "0.0";
      mainAppletContext.lrNewLEARecord.voice = "";
      mainAppletContext.lrNewLEARecord.fax = "";
      mainAppletContext.lrNewLEARecord.DSNVoice = "";
      mainAppletContext.lrNewLEARecord.DSNFax = "";
      mainAppletContext.lrNewLEARecord.email = "";
      mainAppletContext.lrNewLEARecord.url = "";
      mainAppletContext.lrNewLEARecord.password = "";
      mainAppletContext.lrNewLEARecord.strAccessToComputer = "";
      mainAppletContext.lrNewLEARecord.strTypeOfComputer = "Select One";
      mainAppletContext.lrNewLEARecord.strSpeedOfComputer = "Select One";
      mainAppletContext.lrNewLEARecord.strOperatingSystem = "Select One";
      mainAppletContext.lrNewLEARecord.strConnectedToNetwork = "";
      mainAppletContext.lrNewLEARecord.strTypeOfNetworkConnection = "Select One";
      mainAppletContext.lrNewLEARecord.strSpeedOfNetworkConnection = "Select One";
      mainAppletContext.lrNewLEARecord.strBrowser = "";
      mainAppletContext.lrNewLEARecord.strTypeOfBrowser = "Select One";
      mainAppletContext.lrNewLEARecord.strViewedWebPage = "";
      mainAppletContext.strCurrentLoggedInUser = "New LEA";
      mainAppletContext.scrLEAInfoScreen.setCDC(mainAppletContext.strCurrentLoggedInUserCDCU
sername,
                                                mainAppletContext.strCurrentLoggedInUserCDCN
ame);
      mainAppletContext.showScreen(mainAppletContext.scrLEAInfoScreen);
   }
   public void LEAStatisticsClicked(){
      mainAppletContext.wndLEAStatisticsWindow.show();
   }
   public void emailLEAClicked(){
      try{
         mainAppletContext.getAppletContext().showDocument(new URL("mailto:" + mainAppletCon
text.scrLEAInfoScreen.currentRecord.email));
      }
      catch(Exception e){
         e.printStackTrace();
      }
   }
   public void viewHardwareInfoClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEAHardwareScreen);
   }
   public void editHardwareInfoClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEAHardwareScreen);
      if (mainAppletContext.scrLEAHardwareScreen.btnEdit.isVisible()){
         mainAppletContext.scrLEAHardwareScreen.btnEditClicked();
      }
   }
   public void viewOSSoftwareInfoClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEAOSSoftwareScreen);
   }
   public void editOSSoftwareInfoClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEAOSSoftwareScreen);
```

```
      if (mainAppletContext.scrLEAOSSoftwareScreen.btnEdit.isVisible()){
         mainAppletContext.scrLEAOSSoftwareScreen.btnEditClicked();
      }
   }
   public void viewNetworkInfoClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEANetworkScreen);
   }
   public void editNetworkInfoClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEANetworkScreen);
      if (mainAppletContext.scrLEANetworkScreen.btnEdit.isVisible()){
         mainAppletContext.scrLEANetworkScreen.btnEditClicked();
      }
   }
   public void viewWebConnectInfoClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEAWebConnectScreen);
   }
   public void editWebConnectInfoClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEAWebConnectScreen);
      if (mainAppletContext.scrLEAWebConnectScreen.btnEdit.isVisible()){
         mainAppletContext.scrLEAWebConnectScreen.btnEditClicked();
      }
   }

   public void actionPerformed(ActionEvent ae){
      String ac = ae.getActionCommand();
      if (ac.equals("ViewCDCInfo")){
         viewCDCInfoClicked();
      }
      else if (ac.equals("EditCDCInfo")){
         editCDCInfoClicked();
      }
      else if (ac.equals("EmailCDC")){
         emailCDCClicked();
      }
      else if (ac.equals("ViewLEAInfo")){
         viewLEAInfoClicked();
      }
      else if (ac.equals("EditLEAInfo")){
         editLEAInfoClicked();
      }
      else if (ac.equals("AddNewLEA")){
         addNewLEAClicked();
      }
      else if (ac.equals("LEAStatistics")){
         LEAStatisticsClicked();
      }
      else if (ac.equals("EmailLEA")){
         emailLEAClicked();
      }
      else if (ac.equals("ViewHardwareInfo")){
         viewHardwareInfoClicked();
      }
      else if (ac.equals("EditHardwareInfo")){
         editHardwareInfoClicked();
      }
      else if (ac.equals("ViewOSSoftwareInfo")){
         viewOSSoftwareInfoClicked();
      }
      else if (ac.equals("EditOSSoftwareInfo")){
         editOSSoftwareInfoClicked();
      }
      else if (ac.equals("ViewNetworkInfo")){
         viewNetworkInfoClicked();
      }
      else if (ac.equals("EditNetworkInfo")){
```

```
            editNetworkInfoClicked();
        }
        else if (ac.equals("ViewWebConnectInfo")){
            viewWebConnectInfoClicked();
        }
        else if (ac.equals("EditWebConnectInfo")){
            editWebConnectInfoClicked();
        }
    }

    public void showCDCPopup(){
        pmCDC.show(this, 0, 20);
    }
    public void showLEAPopup(){
        pmLEA.show(this, 120, 20);
    }
    public void showHardwarePopup(){
        pmHardware.show(this, 240, 20);
    }
    public void showOSSoftwarePopup(){
        pmOSSoftware.show(this, 360, 20);
    }
    public void showNetworkPopup(){
        pmNetwork.show(this, 480, 20);
    }
    public void showWebConnectPopup(){
        pmWebConnect.show(this, 600, 20);
    }

    public void pushButton(NavigationButton nb){
        nbCDC.buttonSelected = false;
        nbLEA.buttonSelected = false;
        nbHardware.buttonSelected = false;
        nbOSSoftware.buttonSelected = false;
        nbNetwork.buttonSelected = false;
        nbWebConnect.buttonSelected = false;
        nb.buttonSelected = true;
        nbCDC.repaint();
        nbLEA.repaint();
        nbHardware.repaint();
        nbOSSoftware.repaint();
        nbNetwork.repaint();
        nbWebConnect.repaint();
    }

    public void resetButtons(){
        nbCDC.buttonSelected = false;
        nbLEA.buttonSelected = false;
        nbHardware.buttonSelected = false;
        nbOSSoftware.buttonSelected = false;
        nbNetwork.buttonSelected = false;
        nbWebConnect.buttonSelected = false;
        nbCDC.repaint();
        nbLEA.repaint();
        nbHardware.repaint();
        nbOSSoftware.repaint();
        nbNetwork.repaint();
        nbWebConnect.repaint();
    }

    public void pushCDCButton(){
        pushButton(nbCDC);
    }
    public void pushLEAButton(){
        pushButton(nbLEA);
```

```
    }
    public void pushLEAHardwareButton(){
       pushButton(nbHardware);
    }
    public void pushLEAOSSoftwareButton(){
       pushButton(nbOSSoftware);
    }
    public void pushLEANetworkButton(){
       pushButton(nbNetwork);
    }
    public void pushLEAWebConnectButton(){
       pushButton(nbWebConnect);
    }


    public void disableCDCButton(){
       nbCDC.setEnabled(false);
    }

    public void disableLEAButton(){
       nbLEA.setEnabled(false);
    }

    public void disableHardwareButton(){
       nbHardware.setEnabled(false);
    }

    public void disableOSSoftwareButton(){
       nbOSSoftware.setEnabled(false);
    }

    public void disableNetworkButton(){
       nbNetwork.setEnabled(false);
    }

    public void disableWebConnectButton(){
       nbWebConnect.setEnabled(false);
    }

    public void enableCDCButton(){
       nbCDC.setEnabled(true);
    }

    public void enableLEAButton(){
       nbLEA.setEnabled(true);
    }

    public void enableHardwareButton(){
       nbHardware.setEnabled(true);
    }

    public void enableOSSoftwareButton(){
       nbOSSoftware.setEnabled(true);
    }

    public void enableNetworkButton(){
       nbNetwork.setEnabled(true);
    }

    public void enableWebConnectButton(){
       nbWebConnect.setEnabled(true);
    }


    class NavigationButton extends Canvas implements MouseListener{
```

```
    boolean buttonSelected = false;
    int width = 0;
    int height = 0;
    int widthMinus1 = 0;
    int widthMinus2 = 0;
    int heightMinus1 = 0;
    int heightMinus2 = 0;
    FontMetrics fm;
    String strCaption;
    Color textColor;

    NavigationButton(String caption, Font f){
       setBackground(Color.lightGray);
       addMouseListener(this);
       setFont(f);
       strCaption = caption;
    }
    public void setEnabled(boolean enabled){
       super.setEnabled(enabled);
       repaint();
    }
    public void mouseEntered(MouseEvent me){
       textColor = Color.blue;
       repaint();
    }
    public void mouseExited(MouseEvent me){
       textColor = Color.black;
       repaint();
    }
    public void mousePressed(MouseEvent me){}
    public void mouseClicked(MouseEvent me){
       if (isEnabled()){
          //if (! buttonSelected){
             currentlySelectedButton = this;
             update(getGraphics());
          /*}
          else{*/    // show pop up menu
             if (strCaption.equals("CDC Info")){
                showCDCPopup();
             }
             else if (strCaption.equals("LEA Info")){
                showLEAPopup();
             }
             else if (strCaption.equals("Hardware")){
                showHardwarePopup();
             }
             else if (strCaption.equals("OS/Software")){
                showOSSoftwarePopup();
             }
             else if (strCaption.equals("Network")){
                showNetworkPopup();
             }
             else if (strCaption.equals("Web Connect")){
                showWebConnectPopup();
             }
          //}
       }
    }
    public void mouseReleased(MouseEvent me){}
    public void paint(Graphics g){
       width = getSize().width;
       height = getSize().height;
       widthMinus1 = width -1;
       widthMinus2 = width -2;
       heightMinus1 = height -1;
```

```
        heightMinus2 = height -2;
        if (fm == null){
           fm = g.getFontMetrics();
        }
        if (isEnabled()){
           if (buttonSelected){    // button pushed down
              g.setColor(textColor);
              g.drawString(strCaption, (int)( width/2.0 - fm.stringWidth(strCaption)/2.0 +
1.5 ), 16);
              g.setColor(Color.white);
              g.drawLine(0, heightMinus1, widthMinus1, heightMinus1);
              g.drawLine(0, heightMinus2, widthMinus1, heightMinus2);
              g.drawLine(widthMinus1, 0, widthMinus1, heightMinus1);
              g.drawLine(widthMinus2, 0, widthMinus2, heightMinus1);
              g.setColor(Color.black);
              g.drawLine(0, 0, widthMinus1, 0);
              g.setColor(Color.gray);
              g.drawLine(0, 1, widthMinus2, 1);
              g.setColor(Color.black);
              g.drawLine(0, 0, 0, heightMinus1);
              g.setColor(Color.gray);
              g.drawLine(1, 0, 1, heightMinus2);
           }
           else{   // button not pushed down
              g.setColor(textColor);
              g.drawString(strCaption, (int)( width/2.0 - fm.stringWidth(strCaption)/2.0 +
0.5 ), 15);
              g.setColor(Color.white);
              g.drawLine(0, 0, widthMinus1, 0);
              g.drawLine(0, 1, widthMinus1, 1);
              g.drawLine(0, 0, 0, heightMinus1);
              g.drawLine(1, 0, 1, heightMinus1);
              g.setColor(Color.black);
              g.drawLine(0, heightMinus1, widthMinus1, heightMinus1);
              g.setColor(Color.gray);
              g.drawLine(1, heightMinus2, widthMinus1, heightMinus2);
              g.setColor(Color.black);
              g.drawLine(widthMinus1, 0, widthMinus1, heightMinus1);
              g.setColor(Color.gray);
              g.drawLine(widthMinus2, 1, widthMinus2, heightMinus1);
           }
        }
        else{   // draw it as disabled
           g.setColor(Color.white);
           g.drawString(strCaption, (int)( width/2.0 - fm.stringWidth(strCaption)/2.0 + 1.5
), 16);
           g.setColor( new Color(132, 130, 132) );
           g.drawString(strCaption, (int)( width/2.0 - fm.stringWidth(strCaption)/2.0 + 0.5
 ), 15);
           g.setColor(Color.white);
           g.drawLine(0, 0, widthMinus1, 0);
           g.drawLine(0, 1, widthMinus1, 1);
           g.drawLine(0, 0, 0, heightMinus1);
           g.drawLine(1, 0, 1, heightMinus1);
           g.setColor(Color.black);
           g.drawLine(0, heightMinus1, widthMinus1, heightMinus1);
           g.setColor(Color.gray);
           g.drawLine(1, heightMinus2, widthMinus1, heightMinus2);
           g.setColor(Color.black);
           g.drawLine(widthMinus1, 0, widthMinus1, heightMinus1);
           g.setColor(Color.gray);
           g.drawLine(widthMinus2, 1, widthMinus2, heightMinus1);
        }
        g.setColor(Color.black);
     }
```

```
    }

}
```

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;

public class DartAppletMainScreen extends Panel implements ActionListener, MouseListener, Mo
useMotionListener{
   DartApplet mainAppletContext;
   Image backgroundImage = null;
   MouseOverButton btnITCensus, btnCDPlanner, btnDMISpatialIntegrator;
   Polygon plgDartTitleLink;
   Rectangle rctDartImageLink;
   boolean blnMouseInTitleLink = false;
   boolean blnMouseInImageLink = false;

   Image buffer;
   Graphics bufferG;

   public DartAppletMainScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);
      MediaTracker mt = new MediaTracker(this);
      backgroundImage = mainAppletContext.getImage(mainAppletContext.getDocumentBase(), "Bac
kground.jpg");
      mt.addImage(backgroundImage, 0);

      try{
         mt.waitForAll();
      }
      catch(Exception e){
         e.printStackTrace();
      }

      btnITCensus = new MouseOverButton("IT Census");
      btnITCensus.setSize(193, 71);
      btnITCensus.setLocation(132, 276);
      btnITCensus.setFont(mainAppletContext.largeButtonFont);
      btnITCensus.addActionListener(this);
      add(btnITCensus);

      btnCDPlanner = new MouseOverButton("CD Planner");
      btnCDPlanner.setSize(193, 71);
      btnCDPlanner.setLocation(440, 276);
      btnCDPlanner.setFont(mainAppletContext.largeButtonFont);
      btnCDPlanner.addActionListener(this);
      btnCDPlanner.setEnabled(false);
      add(btnCDPlanner);

      btnDMISpatialIntegrator = new MouseOverButton("DMI Spatial Integrator");
      btnDMISpatialIntegrator.setSize(315, 72);
      btnDMISpatialIntegrator.setLocation(225, 391);
      btnDMISpatialIntegrator.setFont(mainAppletContext.largeButtonFont);
      btnDMISpatialIntegrator.addActionListener(this);
      btnDMISpatialIntegrator.setEnabled(false);
      add(btnDMISpatialIntegrator);

      plgDartTitleLink = new Polygon();
      plgDartTitleLink.addPoint(50, 211);
      plgDartTitleLink.addPoint(50, 244);
      plgDartTitleLink.addPoint(686, 244);
      plgDartTitleLink.addPoint(686, 211);
      plgDartTitleLink.addPoint(452, 211);
      plgDartTitleLink.addPoint(452, 151);
      plgDartTitleLink.addPoint(288, 151);
```

```
      plgDartTitleLink.addPoint(288, 211);

      rctDartImageLink = new Rectangle(18, 12, 731, 123);

      addMouseListener(this);
      addMouseMotionListener(this);

   }

   public void setVisible(boolean visible){
      if (visible){
         mainAppletContext.navigationPanel.setVisible(false);
         mainAppletContext.blnInNewLEAMode = false;
         mainAppletContext.navigationPanel.disableLEAButton();
         mainAppletContext.navigationPanel.disableHardwareButton();
         mainAppletContext.navigationPanel.disableOSSoftwareButton();
         mainAppletContext.navigationPanel.disableNetworkButton();
         mainAppletContext.navigationPanel.disableWebConnectButton();
      }
      super.setVisible(visible);
   }

   public void paint(Graphics g){
      if (buffer == null){
         buffer = createImage(getSize().width, getSize().height);
         bufferG = buffer.getGraphics();
      }
      bufferG.drawImage(backgroundImage, 0, 0, null);
      if (blnMouseInTitleLink){
         bufferG.setColor(Color.blue);
         bufferG.drawPolygon(plgDartTitleLink);
      }
      if (blnMouseInImageLink){
         bufferG.setColor(Color.blue);
         bufferG.drawRect(17, 11, 733, 125);
         bufferG.drawRect(16, 10, 735, 127);
      }
      mainAppletContext.drawWindowBorder(bufferG, getSize().width, getSize().height);
      g.drawImage(buffer, 0, 0, null);
   }

   public void update(Graphics g){
      paint(g);
   }

   public void btnITCensusClicked(){
      setVisible(false);
      mainAppletContext.scrLoginScreen.setAdvanceToAndCancelToScreen(mainAppletContext.scrIT
CensusScreen1, this);
      mainAppletContext.scrLoginScreen.setVisible(true);
   }
   public void btnCDPlannerClicked(){
      setVisible(false);
      mainAppletContext.scrLoginScreen.setAdvanceToAndCancelToScreen(mainAppletContext.scrCD
PlannerScreen1, this);
      mainAppletContext.scrLoginScreen.setVisible(true);
   }
   public void btnDMISpatialIntegratorClicked(){
      setVisible(false);
      mainAppletContext.scrLoginScreen.setAdvanceToAndCancelToScreen(mainAppletContext.scrDM
ISpatialIntegratorScreen1, this);
      mainAppletContext.scrLoginScreen.setVisible(true);
   }

   public void actionPerformed(ActionEvent ae){
```

```
      String ac = ae.getActionCommand();
      if (ac.equals("IT Census")){
         btnITCensusClicked();
      }
      else if (ac.equals("CD Planner")){
         btnCDPlannerClicked();
      }
      else if (ac.equals("DMI Spatial Integrator")){
         btnDMISpatialIntegratorClicked();
      }
   }

   public void mouseEntered(MouseEvent me){}
   public void mouseExited(MouseEvent me){}
   public void mousePressed(MouseEvent me){}
   public void mouseClicked(MouseEvent me){
      if (blnMouseInTitleLink){
         setVisible(false);
         mainAppletContext.scrDartInformationScreen.setVisible(true);
      }
      else if (blnMouseInImageLink){
         try{
            mainAppletContext.getAppletContext().showDocument(new URL("http://cdweb.ngb.army
.mil/devnew/"), "_blank");
         }
         catch(Exception e){
            e.printStackTrace();
         }
      }
   }
   public void mouseReleased(MouseEvent me){}
   public void mouseDragged(MouseEvent me){}
   public void mouseMoved(MouseEvent me){
      if (plgDartTitleLink.contains(me.getX(), me.getY())){
         blnMouseInTitleLink = true;
      }
      else{
         blnMouseInTitleLink = false;
      }
      if (rctDartImageLink.contains(me.getX(), me.getY())){
         blnMouseInImageLink = true;
      }
      else{
         blnMouseInImageLink = false;
      }
      getGraphics().setClip(plgDartTitleLink.getBounds().union(rctDartImageLink.getBounds())
);
      update(getGraphics());
      getGraphics().setClip(0, 0, getSize().width, getSize().height);
   }

}
```

```
import java.awt.*;
import java.awt.event.*;

public class DartAppletLoginScreen extends Panel implements ActionListener, ItemListener, Ke
yListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   Label lblDirectionsLine1, lblDirectionsLine2;
   Label lblUsername;
   Label lblPassword;
   TextField txtUsername;
   TextField txtPassword;
   Panel screenToAdvanceTo;
   Panel screenToCancelTo;
   MouseOverButton btnOK, btnCancel, btnYes, btnNo;
   Checkbox chkCDC, chkLEA;
   String strLoginType = "";
   Label lblLEAQuestionLine1, lblLEAQuestionLine2;

   public DartAppletLoginScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setLayout(null);

      lblDirectionsLine1 = new AutoLabel("Please select your type", mainAppletContext.regula
rScreenFont);
      lblDirectionsLine1.setLocation(50, 60);
      add(lblDirectionsLine1);

      lblDirectionsLine2 = new AutoLabel("of counter-drug position:", mainAppletContext.regu
larScreenFont);
      lblDirectionsLine2.setLocation(50, 90);
      add(lblDirectionsLine2);

      chkCDC = new Checkbox("CDC", false);
      chkCDC.setSize(55, 20);
      chkCDC.setLocation(80, 120);
      chkCDC.addItemListener(this);
      chkCDC.setFont(mainAppletContext.regularScreenFont);
      add(chkCDC);

      chkLEA = new Checkbox("LEA", false);
      chkLEA.setSize(55, 20);
      chkLEA.setLocation(140, 120);
      chkLEA.addItemListener(this);
      chkLEA.setFont(mainAppletContext.regularScreenFont);
      add(chkLEA);

      lblUsername = new Label("User Name:");
      lblUsername.setSize(80, 20);
      lblUsername.setLocation(35, 170);
      lblUsername.setFont(mainAppletContext.regularScreenFont);
      add(lblUsername);

      lblPassword = new Label("Password:");
      lblPassword.setSize(80, 20);
      lblPassword.setLocation(35, 210);
      lblPassword.setFont(mainAppletContext.regularScreenFont);
      add(lblPassword);

      txtUsername = new TextField("");
      txtUsername.setSize(100, 20);
      txtUsername.setLocation(120, 170);
      txtUsername.addKeyListener(this);
```

```
        txtUsername.setFont(mainAppletContext.regularScreenFont);
        add(txtUsername);

        txtPassword = new TextField("");
        txtPassword.setSize(100, 20);
        txtPassword.setLocation(120, 210);
        txtPassword.setEchoChar('*');
        txtPassword.addKeyListener(this);
        txtPassword.setFont(mainAppletContext.regularScreenFont);
        add(txtPassword);

        btnOK = new MouseOverButton("OK");
        btnOK.setSize(60, 20);
        btnOK.setLocation(60, 280);
        btnOK.addActionListener(this);
        btnOK.setFont(mainAppletContext.regularScreenFont);
        add(btnOK);

        btnCancel = new MouseOverButton("Cancel");
        btnCancel.setSize(60, 20);
        btnCancel.setLocation(140, 280);
        btnCancel.addActionListener(this);
        btnCancel.setFont(mainAppletContext.regularScreenFont);
        add(btnCancel);

        lblLEAQuestionLine1 = new AutoLabel("Have you ever", mainAppletContext.regularScreenFo
nt);
        lblLEAQuestionLine1.setLocation(80, 180);
        lblLEAQuestionLine1.setVisible(false);
        add(lblLEAQuestionLine1);

        lblLEAQuestionLine2 = new AutoLabel("logged in before?", mainAppletContext.regularScre
enFont);
        lblLEAQuestionLine2.setLocation(70, 210);
        lblLEAQuestionLine2.setVisible(false);
        add(lblLEAQuestionLine2);

        btnYes = new MouseOverButton("Yes");
        btnYes.setSize(60, 20);
        btnYes.setLocation(60, 280);
        btnYes.addActionListener(this);
        btnYes.setFont(mainAppletContext.regularScreenFont);
        btnYes.setVisible(false);
        add(btnYes);

        btnNo = new MouseOverButton("No");
        btnNo.setSize(60, 20);
        btnNo.setLocation(140, 280);
        btnNo.addActionListener(this);
        btnNo.setFont(mainAppletContext.regularScreenFont);
        btnNo.setVisible(false);
        add(btnNo);
    }
    public void paint(Graphics g){
        g.setFont(mainAppletContext.smallerScreenTitleFont);
        fm = g.getFontMetrics();

        tempString = "Login";
        g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

        mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
    }

    public void setVisible(boolean visible){
```

```
      super.setVisible(visible);
      if (visible){
         if ( ! txtPassword.getText().equals("") ){
            txtPassword.setText("");
            txtPassword.requestFocus();
         }
         else{
            chkCDC.setState(false);
            chkLEA.setState(false);
            lblUsername.setVisible(false);
            txtUsername.setVisible(false);
            lblPassword.setVisible(false);
            txtPassword.setVisible(false);
            btnOK.setVisible(false);
            btnCancel.setVisible(false);
            lblLEAQuestionLine1.setVisible(false);
            lblLEAQuestionLine2.setVisible(false);
            btnYes.setVisible(false);
            btnNo.setVisible(false);
            txtUsername.requestFocus();
         }
      }
   }

   public void setAdvanceToAndCancelToScreen(Panel screenToAdvanceTo, Panel screenToCancelTo
){
      this.screenToAdvanceTo = screenToAdvanceTo;
      this.screenToCancelTo = screenToCancelTo;
   }

   public void btnOKClicked(){
      if (! strLoginType.equals("New LEA")){
         Object[] dataTransportArray = new Object[4];
         dataTransportArray[0] = "checkUsernameAndPassword";
         dataTransportArray[1] = strLoginType;
         dataTransportArray[2] = txtUsername.getText();
         dataTransportArray[3] = txtPassword.getText();
         dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
         String[] response = (String[])dataTransportArray[0];
         if ( response[0].equals("accepted") ){
            mainAppletContext.blnInNewLEAMode = false;
            mainAppletContext.strCurrentLoggedInUser = txtUsername.getText();
            mainAppletContext.strCurrentLoggedInUserState = response[1];
            mainAppletContext.strCurrentLoggedInUserCDCUsername = response[2];
            mainAppletContext.strCurrentLoggedInUserCDCName = response[3];
            txtUsername.setText("");
            txtPassword.setText("");
            setVisible(false);
            screenToAdvanceTo.setVisible(true);
         }
         else{
            setVisible(false);
            mainAppletContext.scrLoginFailureScreen.setScreenToGoBackTo(this);
            mainAppletContext.scrLoginFailureScreen.setVisible(true);
         }
      }
   }

   public void btnCancelClicked(){
      txtUsername.setText("");
      txtPassword.setText("");
      setVisible(false);
      screenToCancelTo.setVisible(true);
   }
```

```
    public void btnYesClicked(){
        strLoginType = "LEA";
        lblLEAQuestionLine1.setVisible(false);
        lblLEAQuestionLine2.setVisible(false);
        btnYes.setVisible(false);
        btnNo.setVisible(false);
        lblUsername.setVisible(true);
        lblPassword.setVisible(true);
        txtUsername.setVisible(true);
        txtPassword.setVisible(true);
        btnOK.setVisible(true);
        btnCancel.setVisible(true);
    }

    public void btnNoClicked(){
        strLoginType = "New LEA";
        mainAppletContext.blnInNewLEAMode = true;
        mainAppletContext.blnNewLEAHasGoneThroughScript = false;
        mainAppletContext.lrNewLEARecord = new LEARecord();

        mainAppletContext.lrNewLEARecord.username = "";
        mainAppletContext.lrNewLEARecord.jurisdictionType = "Select One";
        mainAppletContext.lrNewLEARecord.jurisdictionName = "";
        mainAppletContext.lrNewLEARecord.rank = "";
        mainAppletContext.lrNewLEARecord.name = "";
        mainAppletContext.lrNewLEARecord.CDCUsername = "";
        mainAppletContext.lrNewLEARecord.address = "";
        mainAppletContext.lrNewLEARecord.city = "";
        mainAppletContext.lrNewLEARecord.state = "";
        mainAppletContext.lrNewLEARecord.zip = "";
        mainAppletContext.lrNewLEARecord.lat = "0.0";
        mainAppletContext.lrNewLEARecord.lon = "0.0";
        mainAppletContext.lrNewLEARecord.voice = "";
        mainAppletContext.lrNewLEARecord.fax = "";
        mainAppletContext.lrNewLEARecord.DSNVoice = "";
        mainAppletContext.lrNewLEARecord.DSNFax = "";
        mainAppletContext.lrNewLEARecord.email = "";
        mainAppletContext.lrNewLEARecord.url = "";
        mainAppletContext.lrNewLEARecord.password = "";
        mainAppletContext.lrNewLEARecord.strAccessToComputer = "";
        mainAppletContext.lrNewLEARecord.strTypeOfComputer = "Select One";
        mainAppletContext.lrNewLEARecord.strSpeedOfComputer = "Select One";
        mainAppletContext.lrNewLEARecord.strOperatingSystem = "Select One";
        mainAppletContext.lrNewLEARecord.strConnectedToNetwork = "";
        mainAppletContext.lrNewLEARecord.strTypeOfNetworkConnection = "Select One";
        mainAppletContext.lrNewLEARecord.strSpeedOfNetworkConnection = "Select One";
        mainAppletContext.lrNewLEARecord.strBrowser = "";
        mainAppletContext.lrNewLEARecord.strTypeOfBrowser = "Select One";
        mainAppletContext.lrNewLEARecord.strViewedWebPage = "";

        txtUsername.setText("");
        txtPassword.setText("");
        setVisible(false);
        mainAppletContext.strCurrentLoggedInUser = "New LEA";
        mainAppletContext.scrNewLEALoginInformationScreen.setVisible(true);
    }

    public void actionPerformed(ActionEvent ae){
        String ac = ae.getActionCommand();
        if (ac.equals("OK")){
            btnOKClicked();
        }
        else if (ac.equals("Cancel")){
            btnCancelClicked();
        }
```

```
      else if (ac.equals("Yes")){
         btnYesClicked();
      }
      else if (ac.equals("No")){
         btnNoClicked();
      }
   }

   public void chkCDCClicked(){
      strLoginType = "CDC";
      chkLEA.setState(false);
      lblLEAQuestionLine1.setVisible(false);
      lblLEAQuestionLine2.setVisible(false);
      btnYes.setVisible(false);
      btnNo.setVisible(false);
      lblUsername.setVisible(true);
      txtUsername.setVisible(true);
      lblPassword.setVisible(true);
      txtPassword.setVisible(true);
      btnOK.setVisible(true);
      btnCancel.setVisible(true);
   }

   public void chkLEAClicked(){
      chkCDC.setState(false);
      lblLEAQuestionLine1.setVisible(true);
      lblLEAQuestionLine2.setVisible(true);
      btnYes.setVisible(true);
      btnNo.setVisible(true);
      lblUsername.setVisible(false);
      txtUsername.setVisible(false);
      lblPassword.setVisible(false);
      txtPassword.setVisible(false);
      btnOK.setVisible(false);
      btnCancel.setVisible(false);
   }

   public void itemStateChanged(ItemEvent ie){
      Object source = ie.getSource();
      if (source == chkCDC){
         if (chkCDC.getState()){
            chkCDCClicked();
         }
         else{
            chkLEA.setState(true);
            chkLEAClicked();
         }
      }
      else if (source == chkLEA){
         if (chkLEA.getState()){
            chkLEAClicked();
         }
         else{
            chkCDC.setState(true);
            chkCDCClicked();
         }
      }
   }

   public void keyTyped(KeyEvent ke){}
   public void keyReleased(KeyEvent ke){}
   public void keyPressed(KeyEvent ke) {    // why keyPressed and not keyTyped?
      // keyTyped wouldn't work for the enter key
      // "what was the CHARACTER typed" is the idea i think
      // seems odd
```

```
      if (ke.getKeyCode() != KeyEvent.VK_ENTER){
         return;
      }
      else{
         Object source = ke.getSource();
         if (source == txtUsername){
            txtUsername.transferFocus();
         }
         else if (source == txtPassword){
            btnOKClicked();
         }
      }
   }

}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class DartAppletLEAHardwareScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   MouseOverButton btnBack, btnNextPage, btnHome;

   AutoLabel lblLEA, lblJurisdiction;
   TextField txtLEA, txtJurisdiction;

   OutlinedAutoLabel lblQuestion1;
   OutlinedAutoLabel lblQuestion2;
   OutlinedAutoLabel lblQuestion3;

   Checkbox chkAccessYes, chkAccessNo;
   CheckboxGroup cbgAccess;

   Choice chcTypeOfComputer, chcSpeedOfComputer;

   LEARecord currentRecord;

   MouseOverButton btnEdit, btnCancel, btnSave;

   ConfirmEditWindow editConfirmationWindow;

   public DartAppletLEAHardwareScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);

      lblLEA = new AutoLabel("LEA Point of Contact:", mainAppletContext.regularScreenFont);
      lblLEA.setLocation(50, 110);
      add(lblLEA);

      txtLEA = new TextField("");
      txtLEA.setSize(200, 20);
      txtLEA.setLocation(50, 130);
      add(txtLEA);

      lblJurisdiction = new AutoLabel("Jurisdiction:", mainAppletContext.regularScreenFont);
      lblJurisdiction.setLocation(50, 170);
      add(lblJurisdiction);

      txtJurisdiction = new TextField("");
      txtJurisdiction.setSize(200, 20);
      txtJurisdiction.setLocation(50, 190);
      add(txtJurisdiction);

      btnBack = new MouseOverButton("Back");
      btnBack.setSize(80, 20);
      btnBack.setLocation(320, 540);
      btnBack.addActionListener(this);
      btnBack.setFont(mainAppletContext.regularScreenFont);
      add(btnBack);

      btnNextPage = new MouseOverButton("Next Page");
      btnNextPage.setSize(80, 20);
      btnNextPage.setLocation(410, 540);
      btnNextPage.addActionListener(this);
      btnNextPage.setFont(mainAppletContext.regularScreenFont);
      add(btnNextPage);

      btnHome = new MouseOverButton("Home");
```

```
        btnHome.setSize(80, 20);
        btnHome.setLocation(500, 540);
        btnHome.addActionListener(this);
        btnHome.setFont(mainAppletContext.regularScreenFont);
        add(btnHome);

        String[] linesOfText = new String[2];
        linesOfText[0] = "1. Do you have regular access to a computer";
        linesOfText[1] = "to support your LEA duties?";
        lblQuestion1 = new OutlinedAutoLabel(linesOfText, mainAppletContext.smallerScreenTitle
Font);
        lblQuestion1.setLocation(280, 130);
        lblQuestion1.setBackground(Color.white);
        lblQuestion1.setForeground(Color.blue);
        add(lblQuestion1);

        linesOfText[0] = "2. Select the type of computer you";
        linesOfText[1] = "have access to during your LEA duties.";
        lblQuestion2 = new OutlinedAutoLabel(linesOfText, mainAppletContext.smallerScreenTitle
Font);
        lblQuestion2.setLocation(50, 250);
        lblQuestion2.setBackground(Color.white);
        lblQuestion2.setForeground(Color.blue);
        add(lblQuestion2);

        linesOfText[0] = "3. What is the functional level or speed of the";
        linesOfText[1] = "computer used in your LEA duties?";
        lblQuestion3 = new OutlinedAutoLabel(linesOfText, mainAppletContext.smallerScreenTitle
Font);
        lblQuestion3.setLocation(280, 380);
        lblQuestion3.setBackground(Color.white);
        lblQuestion3.setForeground(Color.blue);
        add(lblQuestion3);

        cbgAccess = new CheckboxGroup();

        chkAccessYes = new Checkbox("Yes", false, cbgAccess);
        chkAccessYes.setSize(60, 20);
        chkAccessYes.setLocation(420, 190);
        chkAccessYes.setFont(mainAppletContext.regularScreenFont);
        add(chkAccessYes);

        chkAccessNo = new Checkbox("No", false, cbgAccess);
        chkAccessNo.setSize(60, 20);
        chkAccessNo.setLocation(500, 190);
        chkAccessNo.setFont(mainAppletContext.regularScreenFont);
        add(chkAccessNo);

        chcTypeOfComputer = new Choice();
        chcTypeOfComputer.setSize(300, 20);
        chcTypeOfComputer.setLocation(70, 310);
        chcTypeOfComputer.setFont(mainAppletContext.regularScreenFont);
        chcTypeOfComputer.add("Select One");
        chcTypeOfComputer.add("Common Desktop Personal Computer (PC)");
        chcTypeOfComputer.add("Macintosh");
        chcTypeOfComputer.add("High End Workstation");
        chcTypeOfComputer.add("Internet PC Terminal");
        chcTypeOfComputer.add("Other");
        chcTypeOfComputer.add("None");
        add(chcTypeOfComputer);

        chcSpeedOfComputer = new Choice();
        chcSpeedOfComputer.setSize(100, 20);
        chcSpeedOfComputer.setLocation(420, 440);
        chcSpeedOfComputer.setFont(mainAppletContext.regularScreenFont);
```

```
        chcSpeedOfComputer.add("Select One");
        chcSpeedOfComputer.add("< 200Mhz");
        chcSpeedOfComputer.add(">= 200Mhz");
        chcSpeedOfComputer.add("Other");
        chcSpeedOfComputer.add("None");
        add(chcSpeedOfComputer);

        btnEdit = new MouseOverButton("Edit");
        btnEdit.setSize(60, 20);
        btnEdit.setLocation(694, 150);
        btnEdit.addActionListener(this);
        add(btnEdit);

        btnCancel = new MouseOverButton("Cancel");
        btnCancel.setSize(60, 20);
        btnCancel.setLocation(694, 150);
        btnCancel.addActionListener(this);
        btnCancel.setVisible(false);
        add(btnCancel);

        btnSave = new MouseOverButton("Save");
        btnSave.setSize(60, 20);
        btnSave.setLocation(694, 180);
        btnSave.addActionListener(this);
        btnSave.setVisible(false);
        add(btnSave);

        editConfirmationWindow = new ConfirmEditWindow();
    }

    public void setEnabledState(boolean enabledState){
        chkAccessYes.setEnabled(enabledState);
        chkAccessNo.setEnabled(enabledState);
        chcTypeOfComputer.setEnabled(enabledState);
        chcSpeedOfComputer.setEnabled(enabledState);
    }

    public void setFields(){
        txtLEA.setText(currentRecord.name);
        txtJurisdiction.setText(currentRecord.jurisdictionName);
        if (currentRecord.strAccessToComputer.equals("Yes")){
            cbgAccess.setSelectedCheckbox(chkAccessYes);
        }
        else if (currentRecord.strAccessToComputer.equals("No")){
            cbgAccess.setSelectedCheckbox(chkAccessNo);
        }
        chcTypeOfComputer.select(currentRecord.strTypeOfComputer);
        chcSpeedOfComputer.select(currentRecord.strSpeedOfComputer);
    }

    public void setVisible(boolean visible){
        if (visible){
            currentRecord = mainAppletContext.scrLEAInfoScreen.currentRecord;
            setFields();
            mainAppletContext.navigationPanel.pushLEAHardwareButton();
            setEnabledState(false);
            if( (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.us
ername.toUpperCase())) ||      // only allow lea or his CDC to edit his info
                (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.CD
CUsername.toUpperCase())) ){
                btnEdit.setVisible(true);
            }
            else{
                btnEdit.setVisible(false);
            }
```

```
        }
        super.setVisible(visible);
    }

    public void disableNonEditActionComponents(){
        btnBack.setEnabled(false);
        btnNextPage.setEnabled(false);
        btnHome.setEnabled(false);
        mainAppletContext.navigationPanel.disableCDCButton();
        mainAppletContext.navigationPanel.disableLEAButton();
        mainAppletContext.navigationPanel.disableHardwareButton();
        mainAppletContext.navigationPanel.disableOSSoftwareButton();
        mainAppletContext.navigationPanel.disableNetworkButton();
        mainAppletContext.navigationPanel.disableWebConnectButton();
    }

    public void enableNonEditActionComponents(){
        btnBack.setEnabled(true);
        btnNextPage.setEnabled(true);
        btnHome.setEnabled(true);
        mainAppletContext.navigationPanel.enableCDCButton();
        mainAppletContext.navigationPanel.enableLEAButton();
        mainAppletContext.navigationPanel.enableHardwareButton();
        mainAppletContext.navigationPanel.enableOSSoftwareButton();
        mainAppletContext.navigationPanel.enableNetworkButton();
        mainAppletContext.navigationPanel.enableWebConnectButton();
    }

    public void btnBackClicked(){
        mainAppletContext.showScreen(mainAppletContext.scrLEAInfoScreen);
    }

    public void btnNextPageClicked(){
        if (mainAppletContext.blnInNewLEAMode){
            currentRecord.strAccessToComputer = cbgAccess.getSelectedCheckbox().getLabel();
            currentRecord.strTypeOfComputer = chcTypeOfComputer.getSelectedItem();
            currentRecord.strSpeedOfComputer = chcSpeedOfComputer.getSelectedItem();
        }
        mainAppletContext.showScreen(mainAppletContext.scrLEAOSSoftwareScreen);
    }

    public void btnHomeClicked(){
        mainAppletContext.showScreen(mainAppletContext.scrMainScreen);
    }

    public void btnEditClicked(){
        btnEdit.setVisible(false);
        btnCancel.setVisible(true);
        btnSave.setVisible(true);
        setEnabledState(true);
        disableNonEditActionComponents();
    }

    public void btnCancelClicked(){
        btnCancel.setVisible(false);
        btnSave.setVisible(false);
        btnEdit.setVisible(true);
        setEnabledState(false);
        setFields();   // this will set the textboxes back to the original value
        enableNonEditActionComponents();
    }

    public void btnSaveClicked(){
        editConfirmationWindow.show();
    }
```

```
   public void actionPerformed(ActionEvent ae){
      Object eventSource = ae.getSource();
      if (eventSource == btnBack){
         btnBackClicked();
      }
      else if (eventSource == btnNextPage){
         btnNextPageClicked();
      }
      else if (eventSource == btnHome){
         btnHomeClicked();
      }
      else if (eventSource == btnEdit){
         btnEditClicked();
      }
      else if (eventSource == btnCancel){
         btnCancelClicked();
      }
      else if (eventSource == btnSave){
         btnSaveClicked();
      }
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.screenTitleFont);
      fm = g.getFontMetrics();

      tempString = "LEA Hardware Information";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }

   public void updateRecordAndSendToServer(){
      currentRecord.strAccessToComputer = cbgAccess.getSelectedCheckbox().getLabel();
      currentRecord.strTypeOfComputer = chcTypeOfComputer.getSelectedItem();
      currentRecord.strSpeedOfComputer = chcSpeedOfComputer.getSelectedItem();

      Object[] dataTransportArray = new Object[8];
      dataTransportArray[0] = "updateLEARecord";
      dataTransportArray[1] = currentRecord;
      dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
      enableNonEditActionComponents();
   }

   class ConfirmEditWindow extends Frame implements ActionListener, WindowListener{
      MouseOverButton btnYes;
      MouseOverButton btnNo;
      AutoLabel lblConfirmMessage;
      ConfirmEditWindow(){
         setBackground(Color.lightGray);
         setResizable(false);
         setLayout(null);
         setSize(350, 170);
         setLocation(300, 300);
         setTitle("Save Changes Confirmation");

         lblConfirmMessage = new AutoLabel("Do you really want to save changes?", mainApplet
Context.regularScreenFont);
         lblConfirmMessage.setLocation(50, 60);
         add(lblConfirmMessage);

         btnYes = new MouseOverButton("Yes");
         btnYes.setSize(60, 20);
```

```
        btnYes.setLocation(100, 110);
        btnYes.addActionListener(this);
        add(btnYes);

        btnNo = new MouseOverButton("No");
        btnNo.setSize(60, 20);
        btnNo.setLocation(180, 110);
        btnNo.addActionListener(this);
        add(btnNo);

        addWindowListener(this);
    }
    public void btnYesClicked(){
        // send update to servlet
        updateRecordAndSendToServer();

        setEnabledState(false);
        btnCancel.setVisible(false);
        btnSave.setVisible(false);
        btnEdit.setVisible(true);
        this.setVisible(false);
    }
    public void btnNoClicked(){
        this.setVisible(false);
    }
    public void actionPerformed(ActionEvent ae){
        Object eventSource = ae.getSource();
        if (eventSource == btnYes){
            btnYesClicked();
        }
        else if (eventSource == btnNo){
            btnNoClicked();
        }
    }

    public void windowActivated(WindowEvent we){}
    public void windowClosed(WindowEvent we){}
    public void windowClosing(WindowEvent we){
        this.setVisible(false);
    }
    public void windowDeactivated(WindowEvent we){
        this.setVisible(false);
    }
    public void windowIconified(WindowEvent we){}
    public void windowDeiconified(WindowEvent we){}
    public void windowOpened(WindowEvent we){}

}

}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class DartAppletLEAOSSoftwareScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   MouseOverButton btnBack, btnNextPage, btnHome;

   Label lblLEA, lblJurisdiction;
   TextField txtLEA, txtJurisdiction;

   OutlinedAutoLabel lblQuestion1;

   Choice chcOperatingSystem;

   LEARecord currentRecord;

   MouseOverButton btnEdit, btnCancel, btnSave;

   ConfirmEditWindow editConfirmationWindow;

   public DartAppletLEAOSSoftwareScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);

      lblLEA = new AutoLabel("LEA Point of Contact:", mainAppletContext.regularScreenFont);
      lblLEA.setLocation(50, 110);
      add(lblLEA);

      txtLEA = new TextField("");
      txtLEA.setSize(200, 20);
      txtLEA.setLocation(50, 130);
      add(txtLEA);

      lblJurisdiction = new AutoLabel("Jurisdiction:", mainAppletContext.regularScreenFont);
      lblJurisdiction.setLocation(50, 170);
      add(lblJurisdiction);

      txtJurisdiction = new TextField("");
      txtJurisdiction.setSize(200, 20);
      txtJurisdiction.setLocation(50, 190);
      add(txtJurisdiction);

      btnBack = new MouseOverButton("Back");
      btnBack.setSize(80, 20);
      btnBack.setLocation(320, 540);
      btnBack.addActionListener(this);
      btnBack.setFont(mainAppletContext.regularScreenFont);
      add(btnBack);

      btnNextPage = new MouseOverButton("Next Page");
      btnNextPage.setSize(80, 20);
      btnNextPage.setLocation(410, 540);
      btnNextPage.addActionListener(this);
      btnNextPage.setFont(mainAppletContext.regularScreenFont);
      add(btnNextPage);

      btnHome = new MouseOverButton("Home");
      btnHome.setSize(80, 20);
      btnHome.setLocation(500, 540);
      btnHome.addActionListener(this);
      btnHome.setFont(mainAppletContext.regularScreenFont);
      add(btnHome);
```

```
      lblQuestion1 = new OutlinedAutoLabel("1. What kind of operating system does your compu
ter use?", mainAppletContext.smallerScreenTitleFont);
      lblQuestion1.setLocation(100, 250);
      lblQuestion1.setBackground(Color.white);
      lblQuestion1.setForeground(Color.blue);
      add(lblQuestion1);

      chcOperatingSystem = new Choice();
      chcOperatingSystem.setSize(210, 20);
      chcOperatingSystem.setLocation(245, 290);
      chcOperatingSystem.setFont(mainAppletContext.regularScreenFont);
      chcOperatingSystem.add("Select One");
      chcOperatingSystem.add("MS Windows 95, 98, or 2000");
      chcOperatingSystem.add("Windows NT");
      chcOperatingSystem.add("Macintosh OS");
      chcOperatingSystem.add("UNIX");
      chcOperatingSystem.add("Other");
      chcOperatingSystem.add("None");
      add(chcOperatingSystem);

      btnEdit = new MouseOverButton("Edit");
      btnEdit.setSize(60, 20);
      btnEdit.setLocation(694, 150);
      btnEdit.addActionListener(this);
      add(btnEdit);

      btnCancel = new MouseOverButton("Cancel");
      btnCancel.setSize(60, 20);
      btnCancel.setLocation(694, 150);
      btnCancel.addActionListener(this);
      btnCancel.setVisible(false);
      add(btnCancel);

      btnSave = new MouseOverButton("Save");
      btnSave.setSize(60, 20);
      btnSave.setLocation(694, 180);
      btnSave.addActionListener(this);
      btnSave.setVisible(false);
      add(btnSave);

      editConfirmationWindow = new ConfirmEditWindow();
   }

   public void setEnabledState(boolean enabledState){
      chcOperatingSystem.setEnabled(enabledState);
   }

   public void setFields(){
      txtLEA.setText(currentRecord.name);
      txtJurisdiction.setText(currentRecord.jurisdictionName);
      chcOperatingSystem.select(currentRecord.strOperatingSystem);
   }

   public void setVisible(boolean visible){
      if (visible){
         currentRecord = mainAppletContext.scrLEAInfoScreen.currentRecord;
         setFields();
         mainAppletContext.navigationPanel.pushLEAOSSoftwareButton();
         setEnabledState(false);
         if( (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.us
ername.toUpperCase())) ||      // only allow lea or his CDC to edit his info
             (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.CD
CUsername.toUpperCase())) ){
            btnEdit.setVisible(true);
```

```
        }
        else{
           btnEdit.setVisible(false);
        }
     }
     super.setVisible(visible);
  }

  public void disableNonEditActionComponents(){
     btnBack.setEnabled(false);
     btnNextPage.setEnabled(false);
     btnHome.setEnabled(false);
     mainAppletContext.navigationPanel.disableCDCButton();
     mainAppletContext.navigationPanel.disableLEAButton();
     mainAppletContext.navigationPanel.disableHardwareButton();
     mainAppletContext.navigationPanel.disableOSSoftwareButton();
     mainAppletContext.navigationPanel.disableNetworkButton();
     mainAppletContext.navigationPanel.disableWebConnectButton();
  }

  public void enableNonEditActionComponents(){
     btnBack.setEnabled(true);
     btnNextPage.setEnabled(true);
     btnHome.setEnabled(true);
     mainAppletContext.navigationPanel.enableCDCButton();
     mainAppletContext.navigationPanel.enableLEAButton();
     mainAppletContext.navigationPanel.enableHardwareButton();
     mainAppletContext.navigationPanel.enableOSSoftwareButton();
     mainAppletContext.navigationPanel.enableNetworkButton();
     mainAppletContext.navigationPanel.enableWebConnectButton();
  }

  public void btnBackClicked(){
     mainAppletContext.showScreen(mainAppletContext.scrLEAHardwareScreen);
  }

  public void btnNextPageClicked(){
     if (mainAppletContext.blnInNewLEAMode){
        currentRecord.strOperatingSystem = chcOperatingSystem.getSelectedItem();
     }
     mainAppletContext.showScreen(mainAppletContext.scrLEANetworkScreen);
  }

  public void btnHomeClicked(){
     mainAppletContext.showScreen(mainAppletContext.scrMainScreen);
  }

  public void btnEditClicked(){
     btnEdit.setVisible(false);
     btnCancel.setVisible(true);
     btnSave.setVisible(true);
     setEnabledState(true);
     disableNonEditActionComponents();
  }

  public void btnCancelClicked(){
     btnCancel.setVisible(false);
     btnSave.setVisible(false);
     btnEdit.setVisible(true);
     setEnabledState(false);
     setFields();   // this will set the textboxes back to the original value
     enableNonEditActionComponents();
  }

  public void btnSaveClicked(){
```

```
      editConfirmationWindow.show();
   }

   public void actionPerformed(ActionEvent ae){
      Object eventSource = ae.getSource();
      if (eventSource == btnBack){
         btnBackClicked();
      }
      else if (eventSource == btnNextPage){
         btnNextPageClicked();
      }
      else if (eventSource == btnHome){
         btnHomeClicked();
      }
      else if (eventSource == btnEdit){
         btnEditClicked();
      }
      else if (eventSource == btnCancel){
         btnCancelClicked();
      }
      else if (eventSource == btnSave){
         btnSaveClicked();
      }
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.screenTitleFont);
      fm = g.getFontMetrics();

      tempString = "LEA OS/Software Information";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }

   public void updateRecordAndSendToServer(){
      currentRecord.strOperatingSystem = chcOperatingSystem.getSelectedItem();

      Object[] dataTransportArray = new Object[8];
      dataTransportArray[0] = "updateLEARecord";
      dataTransportArray[1] = currentRecord;
      dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
      enableNonEditActionComponents();
   }

   class ConfirmEditWindow extends Frame implements ActionListener, WindowListener{
      MouseOverButton btnYes;
      MouseOverButton btnNo;
      AutoLabel lblConfirmMessage;
      ConfirmEditWindow(){
         setBackground(Color.lightGray);
         setResizable(false);
         setLayout(null);
         setSize(350, 170);
         setLocation(300, 300);
         setTitle("Save Changes Confirmation");

         lblConfirmMessage = new AutoLabel("Do you really want to save changes?", mainApplet
Context.regularScreenFont);
         lblConfirmMessage.setLocation(50, 60);
         add(lblConfirmMessage);

         btnYes = new MouseOverButton("Yes");
         btnYes.setSize(60, 20);
```

```
        btnYes.setLocation(100, 110);
        btnYes.addActionListener(this);
        add(btnYes);

        btnNo = new MouseOverButton("No");
        btnNo.setSize(60, 20);
        btnNo.setLocation(180, 110);
        btnNo.addActionListener(this);
        add(btnNo);

        addWindowListener(this);
    }
    public void btnYesClicked(){
        // send update to servlet
        updateRecordAndSendToServer();

        setEnabledState(false);
        btnCancel.setVisible(false);
        btnSave.setVisible(false);
        btnEdit.setVisible(true);
        this.setVisible(false);
    }
    public void btnNoClicked(){
        this.setVisible(false);
    }
    public void actionPerformed(ActionEvent ae){
        Object eventSource = ae.getSource();
        if (eventSource == btnYes){
            btnYesClicked();
        }
        else if (eventSource == btnNo){
            btnNoClicked();
        }
    }

    public void windowActivated(WindowEvent we){}
    public void windowClosed(WindowEvent we){}
    public void windowClosing(WindowEvent we){
        this.setVisible(false);
    }
    public void windowDeactivated(WindowEvent we){
        this.setVisible(false);
    }
    public void windowIconified(WindowEvent we){}
    public void windowDeiconified(WindowEvent we){}
    public void windowOpened(WindowEvent we){}

  }

}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class DartAppletLEANetworkScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   MouseOverButton btnBack, btnNextPage, btnHome;

   Label lblLEA, lblJurisdiction;
   TextField txtLEA, txtJurisdiction;

   OutlinedAutoLabel lblQuestion1, lblQuestion2, lblQuestion3;

   Checkbox chkNetworkConnectionYes, chkNetworkConnectionNo;
   CheckboxGroup cbgNetworkConnection;

   Choice chcTypeOfNetworkConnection, chcSpeedOfNetworkConnection;

   LEARecord currentRecord;

   MouseOverButton btnEdit, btnCancel, btnSave;

   ConfirmEditWindow editConfirmationWindow;

   public DartAppletLEANetworkScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);

      lblLEA = new AutoLabel("LEA Point of Contact:", mainAppletContext.regularScreenFont);
      lblLEA.setLocation(50, 110);
      add(lblLEA);

      txtLEA = new TextField("");
      txtLEA.setSize(200, 20);
      txtLEA.setLocation(50, 130);
      add(txtLEA);

      lblJurisdiction = new AutoLabel("Jurisdiction:", mainAppletContext.regularScreenFont);
      lblJurisdiction.setLocation(50, 170);
      add(lblJurisdiction);

      txtJurisdiction = new TextField("");
      txtJurisdiction.setSize(200, 20);
      txtJurisdiction.setLocation(50, 190);
      add(txtJurisdiction);

      btnBack = new MouseOverButton("Back");
      btnBack.setSize(80, 20);
      btnBack.setLocation(320, 540);
      btnBack.addActionListener(this);
      btnBack.setFont(mainAppletContext.regularScreenFont);
      add(btnBack);

      btnNextPage = new MouseOverButton("Next Page");
      btnNextPage.setSize(80, 20);
      btnNextPage.setLocation(410, 540);
      btnNextPage.addActionListener(this);
      btnNextPage.setFont(mainAppletContext.regularScreenFont);
      add(btnNextPage);

      btnHome = new MouseOverButton("Home");
      btnHome.setSize(80, 20);
      btnHome.setLocation(500, 540);
```

```
        btnHome.addActionListener(this);
        btnHome.setFont(mainAppletContext.regularScreenFont);
        add(btnHome);

        lblQuestion1 = new OutlinedAutoLabel("1. Is this computer connected to a network?", ma
inAppletContext.smallerScreenTitleFont);
        lblQuestion1.setLocation(280, 130);
        lblQuestion1.setBackground(Color.white);
        lblQuestion1.setForeground(Color.blue);
        add(lblQuestion1);

        lblQuestion2 = new OutlinedAutoLabel("2. What kind of network connection do you have?"
, mainAppletContext.smallerScreenTitleFont);
        lblQuestion2.setLocation(50, 250);
        lblQuestion2.setBackground(Color.white);
        lblQuestion2.setForeground(Color.blue);
        add(lblQuestion2);

        lblQuestion3 = new OutlinedAutoLabel("3. What speed is your network connection?", main
AppletContext.smallerScreenTitleFont);
        lblQuestion3.setLocation(280, 380);
        lblQuestion3.setBackground(Color.white);
        lblQuestion3.setForeground(Color.blue);
        add(lblQuestion3);

        cbgNetworkConnection = new CheckboxGroup();

        chkNetworkConnectionYes = new Checkbox("Yes", false, cbgNetworkConnection);
        chkNetworkConnectionYes.setSize(60, 20);
        chkNetworkConnectionYes.setLocation(420, 170);
        chkNetworkConnectionYes.setFont(mainAppletContext.regularScreenFont);
        add(chkNetworkConnectionYes);

        chkNetworkConnectionNo = new Checkbox("No", false, cbgNetworkConnection);
        chkNetworkConnectionNo.setSize(60, 20);
        chkNetworkConnectionNo.setLocation(500, 170);
        chkNetworkConnectionNo.setFont(mainAppletContext.regularScreenFont);
        add(chkNetworkConnectionNo);

        chcTypeOfNetworkConnection = new Choice();
        chcTypeOfNetworkConnection.setSize(200, 20);
        chcTypeOfNetworkConnection.setLocation(150, 290);
        chcTypeOfNetworkConnection.setFont(mainAppletContext.regularScreenFont);
        chcTypeOfNetworkConnection.add("Select One");
        chcTypeOfNetworkConnection.add("Modem over Phone Line");
        chcTypeOfNetworkConnection.add("Local Router");
        chcTypeOfNetworkConnection.add("High Speed Multiplexer");
        chcTypeOfNetworkConnection.add("Unknown");
        chcTypeOfNetworkConnection.add("Other");
        chcTypeOfNetworkConnection.add("None");
        add(chcTypeOfNetworkConnection);

        chcSpeedOfNetworkConnection = new Choice();
        chcSpeedOfNetworkConnection.setSize(200, 20);
        chcSpeedOfNetworkConnection.setLocation(370, 420);
        chcSpeedOfNetworkConnection.setFont(mainAppletContext.regularScreenFont);
        chcSpeedOfNetworkConnection.add("Select One");
        chcSpeedOfNetworkConnection.add("56K or greater");
        chcSpeedOfNetworkConnection.add("T1 or greater");
        chcSpeedOfNetworkConnection.add("Unknown");
        chcSpeedOfNetworkConnection.add("Other");
        chcSpeedOfNetworkConnection.add("None");
        add(chcSpeedOfNetworkConnection);

        btnEdit = new MouseOverButton("Edit");
```

```
      btnEdit.setSize(60, 20);
      btnEdit.setLocation(694, 150);
      btnEdit.addActionListener(this);
      add(btnEdit);

      btnCancel = new MouseOverButton("Cancel");
      btnCancel.setSize(60, 20);
      btnCancel.setLocation(694, 150);
      btnCancel.addActionListener(this);
      btnCancel.setVisible(false);
      add(btnCancel);

      btnSave = new MouseOverButton("Save");
      btnSave.setSize(60, 20);
      btnSave.setLocation(694, 180);
      btnSave.addActionListener(this);
      btnSave.setVisible(false);
      add(btnSave);

      editConfirmationWindow = new ConfirmEditWindow();
   }

   public void setEnabledState(boolean enabledState){
      chkNetworkConnectionYes.setEnabled(enabledState);
      chkNetworkConnectionNo.setEnabled(enabledState);
      chcTypeOfNetworkConnection.setEnabled(enabledState);
      chcSpeedOfNetworkConnection.setEnabled(enabledState);
   }

   public void setFields(){
      txtLEA.setText(currentRecord.name);
      txtJurisdiction.setText(currentRecord.jurisdictionName);
      if (currentRecord.strConnectedToNetwork.equals("Yes")){
         cbgNetworkConnection.setSelectedCheckbox(chkNetworkConnectionYes);
      }
      else if (currentRecord.strConnectedToNetwork.equals("No")){
         cbgNetworkConnection.setSelectedCheckbox(chkNetworkConnectionNo);
      }
      chcTypeOfNetworkConnection.select(currentRecord.strTypeOfNetworkConnection);
      chcSpeedOfNetworkConnection.select(currentRecord.strSpeedOfNetworkConnection);
   }

   public void setVisible(boolean visible){
      if (visible){
         currentRecord = mainAppletContext.scrLEAInfoScreen.currentRecord;
         setFields();
         mainAppletContext.navigationPanel.pushLEANetworkButton();
         setEnabledState(false);
         if( (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.us
ername.toUpperCase())) ||      // only allow lea or his CDC to edit his info
             (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.CD
CUsername.toUpperCase())) ){
            btnEdit.setVisible(true);
         }
         else{
            btnEdit.setVisible(false);
         }
      }
      super.setVisible(visible);
   }

   public void disableNonEditActionComponents(){
      btnBack.setEnabled(false);
      btnNextPage.setEnabled(false);
      btnHome.setEnabled(false);
```

```
      mainAppletContext.navigationPanel.disableCDCButton();
      mainAppletContext.navigationPanel.disableLEAButton();
      mainAppletContext.navigationPanel.disableHardwareButton();
      mainAppletContext.navigationPanel.disableOSSoftwareButton();
      mainAppletContext.navigationPanel.disableNetworkButton();
      mainAppletContext.navigationPanel.disableWebConnectButton();
   }

   public void enableNonEditActionComponents(){
      btnBack.setEnabled(true);
      btnNextPage.setEnabled(true);
      btnHome.setEnabled(true);
      mainAppletContext.navigationPanel.enableCDCButton();
      mainAppletContext.navigationPanel.enableLEAButton();
      mainAppletContext.navigationPanel.enableHardwareButton();
      mainAppletContext.navigationPanel.enableOSSoftwareButton();
      mainAppletContext.navigationPanel.enableNetworkButton();
      mainAppletContext.navigationPanel.enableWebConnectButton();
   }

   public void btnBackClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEAOSSoftwareScreen);
   }

   public void btnNextPageClicked(){
      if (mainAppletContext.blnInNewLEAMode){
         currentRecord.strConnectedToNetwork = cbgNetworkConnection.getSelectedCheckbox().ge
tLabel();
         currentRecord.strTypeOfNetworkConnection = chcTypeOfNetworkConnection.getSelectedIt
em();
         currentRecord.strSpeedOfNetworkConnection = chcSpeedOfNetworkConnection.getSelected
Item();
      }
      mainAppletContext.showScreen(mainAppletContext.scrLEAWebConnectScreen);
   }

   public void btnHomeClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrMainScreen);
   }

   public void btnEditClicked(){
      btnEdit.setVisible(false);
      btnCancel.setVisible(true);
      btnSave.setVisible(true);
      setEnabledState(true);
      disableNonEditActionComponents();
   }

   public void btnCancelClicked(){
      btnCancel.setVisible(false);
      btnSave.setVisible(false);
      btnEdit.setVisible(true);
      setEnabledState(false);
      setFields();   // this will set the textboxes back to the original value
      enableNonEditActionComponents();
   }

   public void btnSaveClicked(){
      editConfirmationWindow.show();
   }

   public void actionPerformed(ActionEvent ae){
      Object eventSource = ae.getSource();
      if (eventSource == btnBack){
         btnBackClicked();
```

```
      }
      else if (eventSource == btnNextPage){
         btnNextPageClicked();
      }
      else if (eventSource == btnHome){
         btnHomeClicked();
      }
      else if (eventSource == btnEdit){
         btnEditClicked();
      }
      else if (eventSource == btnCancel){
         btnCancelClicked();
      }
      else if (eventSource == btnSave){
         btnSaveClicked();
      }
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.screenTitleFont);
      fm = g.getFontMetrics();

      tempString = "LEA Network Information";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }

   public void updateRecordAndSendToServer(){
      currentRecord.strConnectedToNetwork = cbgNetworkConnection.getSelectedCheckbox().getLa
bel();
      currentRecord.strTypeOfNetworkConnection = chcTypeOfNetworkConnection.getSelectedItem(
);
      currentRecord.strSpeedOfNetworkConnection = chcSpeedOfNetworkConnection.getSelectedIte
m();

      Object[] dataTransportArray = new Object[8];
      dataTransportArray[0] = "updateLEARecord";
      dataTransportArray[1] = currentRecord;
      dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
      enableNonEditActionComponents();
   }

   class ConfirmEditWindow extends Frame implements ActionListener, WindowListener{
      MouseOverButton btnYes;
      MouseOverButton btnNo;
      AutoLabel lblConfirmMessage;
      ConfirmEditWindow(){
         setBackground(Color.lightGray);
         setResizable(false);
         setLayout(null);
         setSize(350, 170);
         setLocation(300, 300);
         setTitle("Save Changes Confirmation");

         lblConfirmMessage = new AutoLabel("Do you really want to save changes?", mainApplet
Context.regularScreenFont);
         lblConfirmMessage.setLocation(50, 60);
         add(lblConfirmMessage);

         btnYes = new MouseOverButton("Yes");
         btnYes.setSize(60, 20);
         btnYes.setLocation(100, 110);
         btnYes.addActionListener(this);
```

```
        add(btnYes);

        btnNo = new MouseOverButton("No");
        btnNo.setSize(60, 20);
        btnNo.setLocation(180, 110);
        btnNo.addActionListener(this);
        add(btnNo);

        addWindowListener(this);
    }
    public void btnYesClicked(){
        // send update to servlet
        updateRecordAndSendToServer();

        setEnabledState(false);
        btnCancel.setVisible(false);
        btnSave.setVisible(false);
        btnEdit.setVisible(true);
        this.setVisible(false);
    }
    public void btnNoClicked(){
        this.setVisible(false);
    }
    public void actionPerformed(ActionEvent ae){
        Object eventSource = ae.getSource();
        if (eventSource == btnYes){
            btnYesClicked();
        }
        else if (eventSource == btnNo){
            btnNoClicked();
        }
    }

    public void windowActivated(WindowEvent we){}
    public void windowClosed(WindowEvent we){}
    public void windowClosing(WindowEvent we){
        this.setVisible(false);
    }
    public void windowDeactivated(WindowEvent we){
        this.setVisible(false);
    }
    public void windowIconified(WindowEvent we){}
    public void windowDeiconified(WindowEvent we){}
    public void windowOpened(WindowEvent we){}

}

}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class DartAppletLEAWebConnectScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   MouseOverButton btnBack, btnNextPage, btnHome;

   Label lblLEA, lblJurisdiction;
   TextField txtLEA, txtJurisdiction;

   OutlinedAutoLabel lblQuestion1, lblQuestion2, lblQuestion3;

   Checkbox chkBrowserYes, chkBrowserNo;
   CheckboxGroup cbgBrowser;

   Choice chcBrowserName;

   Checkbox chkViewedWebPageYes, chkViewedWebPageNo;
   CheckboxGroup cbgViewedWebPage;

   LEARecord currentRecord;

   MouseOverButton btnEdit, btnCancel, btnSave;

   ConfirmEditWindow editConfirmationWindow;

   public DartAppletLEAWebConnectScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);

      lblLEA = new AutoLabel("LEA Point of Contact:", mainAppletContext.regularScreenFont);
      lblLEA.setLocation(50, 110);
      add(lblLEA);

      txtLEA = new TextField("");
      txtLEA.setSize(200, 20);
      txtLEA.setLocation(50, 130);
      add(txtLEA);

      lblJurisdiction = new AutoLabel("Jurisdiction:", mainAppletContext.regularScreenFont);
      lblJurisdiction.setLocation(50, 170);
      add(lblJurisdiction);

      txtJurisdiction = new TextField("");
      txtJurisdiction.setSize(200, 20);
      txtJurisdiction.setLocation(50, 190);
      add(txtJurisdiction);

      btnBack = new MouseOverButton("Back");
      btnBack.setSize(80, 20);
      btnBack.setLocation(320, 540);
      btnBack.addActionListener(this);
      btnBack.setFont(mainAppletContext.regularScreenFont);
      add(btnBack);

      btnNextPage = new MouseOverButton("Next Page");
      btnNextPage.setSize(80, 20);
      btnNextPage.setLocation(410, 540);
      btnNextPage.addActionListener(this);
      btnNextPage.setEnabled(false);
      btnNextPage.setFont(mainAppletContext.regularScreenFont);
      add(btnNextPage);
```

```
        btnHome = new MouseOverButton("Home");
        btnHome.setSize(80, 20);
        btnHome.setLocation(500, 540);
        btnHome.addActionListener(this);
        btnHome.setFont(mainAppletContext.regularScreenFont);
        add(btnHome);

        String[] linesOfText = new String[2];
        linesOfText[0] = "1. Does this computer have browser software";
        linesOfText[1] = "to connect to the world wide web?";
        lblQuestion1 = new OutlinedAutoLabel(linesOfText, mainAppletContext.smallerScreenTitle
Font);
        lblQuestion1.setLocation(280, 130);
        lblQuestion1.setBackground(Color.white);
        lblQuestion1.setForeground(Color.blue);
        add(lblQuestion1);

        lblQuestion2 = new OutlinedAutoLabel("2. What kind of browser software do you have?",
mainAppletContext.smallerScreenTitleFont);
        lblQuestion2.setLocation(50, 250);
        lblQuestion2.setBackground(Color.white);
        lblQuestion2.setForeground(Color.blue);
        add(lblQuestion2);

        lblQuestion3 = new OutlinedAutoLabel("3. Have you ever viewed a webpage using this com
puter?",
                mainAppletContext.smallerScreenTitleFont);
        lblQuestion3.setLocation(250, 380);
        lblQuestion3.setBackground(Color.white);
        lblQuestion3.setForeground(Color.blue);
        add(lblQuestion3);

        cbgBrowser = new CheckboxGroup();

        chkBrowserYes = new Checkbox("Yes", false, cbgBrowser);
        chkBrowserYes.setSize(60, 20);
        chkBrowserYes.setLocation(420, 190);
        chkBrowserYes.setFont(mainAppletContext.regularScreenFont);
        add(chkBrowserYes);

        chkBrowserNo = new Checkbox("No", false, cbgBrowser);
        chkBrowserNo.setSize(60, 20);
        chkBrowserNo.setLocation(500, 190);
        chkBrowserNo.setFont(mainAppletContext.regularScreenFont);
        add(chkBrowserNo);

        chcBrowserName = new Choice();
        chcBrowserName.setSize(200, 20);
        chcBrowserName.setLocation(150, 290);
        chcBrowserName.setFont(mainAppletContext.regularScreenFont);
        chcBrowserName.add("Select One");
        chcBrowserName.add("Netscape Navigator");
        chcBrowserName.add("MS Internet Explorer");
        chcBrowserName.add("Unknown");
        chcBrowserName.add("Other");
        chcBrowserName.add("None");
        add(chcBrowserName);

        cbgViewedWebPage = new CheckboxGroup();

        chkViewedWebPageYes = new Checkbox("Yes", true, cbgViewedWebPage);
        chkViewedWebPageYes.setSize(60, 20);
        chkViewedWebPageYes.setLocation(430, 420);
        chkViewedWebPageYes.setFont(mainAppletContext.regularScreenFont);
        add(chkViewedWebPageYes);
```

```
        chkViewedWebPageNo = new Checkbox("No", false, cbgViewedWebPage);
        chkViewedWebPageNo.setSize(60, 20);
        chkViewedWebPageNo.setLocation(510, 420);
        chkViewedWebPageNo.setFont(mainAppletContext.regularScreenFont);
        add(chkViewedWebPageNo);

        btnEdit = new MouseOverButton("Edit");
        btnEdit.setSize(60, 20);
        btnEdit.setLocation(694, 150);
        btnEdit.addActionListener(this);
        add(btnEdit);

        btnCancel = new MouseOverButton("Cancel");
        btnCancel.setSize(60, 20);
        btnCancel.setLocation(694, 150);
        btnCancel.addActionListener(this);
        btnCancel.setVisible(false);
        add(btnCancel);

        btnSave = new MouseOverButton("Save");
        btnSave.setSize(60, 20);
        btnSave.setLocation(694, 180);
        btnSave.addActionListener(this);
        btnSave.setVisible(false);
        add(btnSave);

        editConfirmationWindow = new ConfirmEditWindow();
    }

    public void setEnabledState(boolean enabledState){
        chkBrowserYes.setEnabled(enabledState);
        chkBrowserNo.setEnabled(enabledState);
        chcBrowserName.setEnabled(enabledState);
        chkViewedWebPageYes.setEnabled(enabledState);
        chkViewedWebPageNo.setEnabled(enabledState);
    }

    public void setFields(){
        txtLEA.setText(currentRecord.name);
        txtJurisdiction.setText(currentRecord.jurisdictionName);
        if (currentRecord.strBrowser.equals("Yes")){
            cbgBrowser.setSelectedCheckbox(chkBrowserYes);
        }
        else if (currentRecord.strBrowser.equals("No")){
            cbgBrowser.setSelectedCheckbox(chkBrowserNo);
        }
        chcBrowserName.select(currentRecord.strTypeOfBrowser);
        if (currentRecord.strViewedWebPage.equals("Yes")){
            cbgViewedWebPage.setSelectedCheckbox(chkViewedWebPageYes);
        }
        else if (currentRecord.strViewedWebPage.equals("No")){
            cbgViewedWebPage.setSelectedCheckbox(chkViewedWebPageNo);
        }
    }

    public void setVisible(boolean visible){
        if (visible){
            currentRecord = mainAppletContext.scrLEAInfoScreen.currentRecord;
            setFields();
            mainAppletContext.navigationPanel.pushLEAWebConnectButton();
            setEnabledState(false);
            if( (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.us
ername.toUpperCase())) ||      // only allow lea or his CDC to edit his info
                (mainAppletContext.strCurrentLoggedInUser.toUpperCase().equals(currentRecord.CD
```

```
CUsername.toUpperCase())) ){
            btnEdit.setVisible(true);
         }
         else{
            btnEdit.setVisible(false);
         }
      }
      super.setVisible(visible);
   }

   public void disableNonEditActionComponents(){
      btnBack.setEnabled(false);
      btnNextPage.setEnabled(false);
      btnHome.setEnabled(false);
      mainAppletContext.navigationPanel.disableCDCButton();
      mainAppletContext.navigationPanel.disableLEAButton();
      mainAppletContext.navigationPanel.disableHardwareButton();
      mainAppletContext.navigationPanel.disableOSSoftwareButton();
      mainAppletContext.navigationPanel.disableNetworkButton();
      mainAppletContext.navigationPanel.disableWebConnectButton();
   }

   public void enableNonEditActionComponents(){
      btnBack.setEnabled(true);
      btnNextPage.setEnabled(false);
      btnHome.setEnabled(true);
      mainAppletContext.navigationPanel.enableCDCButton();
      mainAppletContext.navigationPanel.enableLEAButton();
      mainAppletContext.navigationPanel.enableHardwareButton();
      mainAppletContext.navigationPanel.enableOSSoftwareButton();
      mainAppletContext.navigationPanel.enableNetworkButton();
      mainAppletContext.navigationPanel.enableWebConnectButton();
   }

   public void btnBackClicked(){
      if (mainAppletContext.blnInNewLEAMode){
         currentRecord.strBrowser = cbgBrowser.getSelectedCheckbox().getLabel();
         currentRecord.strTypeOfBrowser = chcBrowserName.getSelectedItem();
         currentRecord.strViewedWebPage = cbgViewedWebPage.getSelectedCheckbox().getLabel();
      }
      mainAppletContext.showScreen(mainAppletContext.scrLEANetworkScreen);
   }

   public void btnNextPageClicked(){
   }

   public void btnHomeClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrMainScreen);
   }

   public void btnEditClicked(){
      btnEdit.setVisible(false);
      btnCancel.setVisible(true);
      btnSave.setVisible(true);
      setEnabledState(true);
      disableNonEditActionComponents();
   }

   public void btnCancelClicked(){
      btnCancel.setVisible(false);
      btnSave.setVisible(false);
      btnEdit.setVisible(true);
      setEnabledState(false);
      setFields();   // this will set the textboxes back to the original value
      enableNonEditActionComponents();
```

```
    }

    public void btnSaveClicked(){
        editConfirmationWindow.show();
    }

    public void actionPerformed(ActionEvent ae){
        Object eventSource = ae.getSource();
        if (eventSource == btnBack){
            btnBackClicked();
        }
        else if (eventSource == btnNextPage){
            btnNextPageClicked();
        }
        else if (eventSource == btnHome){
            btnHomeClicked();
        }
        else if (eventSource == btnEdit){
            btnEditClicked();
        }
        else if (eventSource == btnCancel){
            btnCancelClicked();
        }
        else if (eventSource == btnSave){
            btnSaveClicked();
        }
    }

    public void paint(Graphics g){
        g.setFont(mainAppletContext.screenTitleFont);
        fm = g.getFontMetrics();

        tempString = "LEA WebConnect Information";
        g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

        mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
    }

    public void updateRecordAndSendToServer(){
        currentRecord.strBrowser = cbgBrowser.getSelectedCheckbox().getLabel();
        currentRecord.strTypeOfBrowser = chcBrowserName.getSelectedItem();
        currentRecord.strViewedWebPage = cbgViewedWebPage.getSelectedCheckbox().getLabel();

        Object[] dataTransportArray = new Object[8];
        dataTransportArray[0] = "updateLEARecord";
        dataTransportArray[1] = currentRecord;
        dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
        enableNonEditActionComponents();
    }

    class ConfirmEditWindow extends Frame implements ActionListener, WindowListener{
        MouseOverButton btnYes;
        MouseOverButton btnNo;
        AutoLabel lblConfirmMessage;
        ConfirmEditWindow(){
            setBackground(Color.lightGray);
            setResizable(false);
            setLayout(null);
            setSize(350, 170);
            setLocation(300, 300);
            setTitle("Save Changes Confirmation");

            lblConfirmMessage = new AutoLabel("Do you really want to save changes?", mainApplet
Context.regularScreenFont);
```

```
        lblConfirmMessage.setLocation(50, 60);
        add(lblConfirmMessage);

        btnYes = new MouseOverButton("Yes");
        btnYes.setSize(60, 20);
        btnYes.setLocation(100, 110);
        btnYes.addActionListener(this);
        add(btnYes);

        btnNo = new MouseOverButton("No");
        btnNo.setSize(60, 20);
        btnNo.setLocation(180, 110);
        btnNo.addActionListener(this);
        add(btnNo);

        addWindowListener(this);
    }
    public void btnYesClicked(){
        // send update to servlet
        updateRecordAndSendToServer();

        setEnabledState(false);
        btnCancel.setVisible(false);
        btnSave.setVisible(false);
        btnEdit.setVisible(true);
        this.setVisible(false);
    }
    public void btnNoClicked(){
        this.setVisible(false);
    }
    public void actionPerformed(ActionEvent ae){
        Object eventSource = ae.getSource();
        if (eventSource == btnYes){
            btnYesClicked();
        }
        else if (eventSource == btnNo){
            btnNoClicked();
        }
    }

    public void windowActivated(WindowEvent we){}
    public void windowClosed(WindowEvent we){}
    public void windowClosing(WindowEvent we){
        this.setVisible(false);
    }
    public void windowDeactivated(WindowEvent we){
        this.setVisible(false);
    }
    public void windowIconified(WindowEvent we){}
    public void windowDeiconified(WindowEvent we){}
    public void windowOpened(WindowEvent we){}

}

}
```

```
import java.awt.*;

class FlashingPrompt extends Canvas{
   Image buffer;
   Graphics bufferG;
   Polygon leftPolygon, rightPolygon, upPolygon, downPolygon, selectedPolygon;
   Color flashColor = new Color(0, 0, 0);
   public final int RIGHT = 0;
   public final int LEFT = 1;
   String flashDirection = "right";
   FlashThread ft;
   public void startFlashing(){
      ft = new FlashThread();
      ft.start();
   }
   public void stopFlashing(){
      if (ft != null){
         ft.die();
      }
   }
   public void setSize(int width, int height){
      super.setSize(width, height);
      leftPolygon = new Polygon();
      leftPolygon.addPoint(0, (int)(getSize().height/2.0 + 0.5) );
      leftPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), 0);
      leftPolygon.addPoint( getSize().width, 0);
      leftPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), (int)(getSize().height/2.0 + 0
.5) );
      leftPolygon.addPoint( getSize().width, getSize().height);
      leftPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), getSize().height);

      rightPolygon = new Polygon();
      rightPolygon.addPoint(0, 0);
      rightPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), 0);
      rightPolygon.addPoint( getSize().width, (int)(getSize().height/2.0 + 0.5) );
      rightPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), getSize().height);
      rightPolygon.addPoint(0, getSize().height);
      rightPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), (int)(getSize().height/2.0 +
0.5) );

      upPolygon = new Polygon();
      upPolygon.addPoint(0, getSize().height);
      upPolygon.addPoint(0, (int)(getSize().height/2.0 + 0.5));
      upPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), 0);
      upPolygon.addPoint( getSize().width, (int)(getSize().height/2.0 + 0.5));
      upPolygon.addPoint(getSize().width, getSize().height);
      upPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), (int)(getSize().height/2.0 + 0.5
) );

      downPolygon = new Polygon();
      downPolygon.addPoint(0, 0);
      downPolygon.addPoint( 0, (int)(getSize().height/2.0 + 0.5));
      downPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), getSize().height);
      downPolygon.addPoint( getSize().width, (int)(getSize().height/2.0 + 0.5));
      downPolygon.addPoint(getSize().width, 0);
      downPolygon.addPoint( (int)(getSize().width/2.0 + 0.5), (int)(getSize().height/2.0 + 0
.5) );
   }
   public void setDirection(String direction){
      flashDirection = direction;
      repaint();
   }
   public void paint(Graphics g){
      if (buffer == null){
         buffer = createImage(getSize().width, getSize().height);
```

```
            bufferG = buffer.getGraphics();
        }
        bufferG.setColor(getBackground());
        bufferG.fillRect(0, 0, getSize().width, getSize().height);
        if (flashDirection.equals("right")){
            selectedPolygon = rightPolygon;
        }
        else if (flashDirection.equals("left")){
            selectedPolygon = leftPolygon;
        }
        else if (flashDirection.equals("up")){
            selectedPolygon = upPolygon;
        }
        else if (flashDirection.equals("down")){
            selectedPolygon = downPolygon;
        }
        bufferG.setColor(flashColor);
        bufferG.fillPolygon(selectedPolygon);
        g.drawImage(buffer, 0, 0, null);
    }
    public void update(Graphics g){
        paint(g);
    }
    class FlashThread extends Thread{
        int red, green, blue;
        boolean blnBrighten = true;
        boolean blnStillLiving = true;
        public void die(){
            blnStillLiving = false;
        }
        public void run(){
            while(true && blnStillLiving){
                try{
                    sleep(10);
                }
                catch(Exception e){
                    e.printStackTrace();
                }
                red = flashColor.getRed();
                green = flashColor.getGreen();
                blue = flashColor.getBlue();
                if ( (red >= 253) || (green == 255) || (blue == 255) ){
                    blnBrighten = false;
                }
                else if ( (red <= 3) /*|| (green == 0) || (blue == 0)*/ ){
                    blnBrighten = true;
                }
                if (blnBrighten){
                    flashColor = new Color(flashColor.getRed()+3, flashColor.getGreen(), flashCol
or.getBlue());
                }
                else{
                    flashColor = new Color(flashColor.getRed()-3, flashColor.getGreen(), flashCol
or.getBlue());
                }
                repaint();
            }
        }
    }

}
```

```
import java.awt.*;
import java.awt.event.*;

public class MouseOverButton extends Button implements MouseListener{
   MouseOverButton(){
      super();
      addMouseListener(this);
   }
   MouseOverButton(String caption){
      super(caption);
      addMouseListener(this);
   }
   public void mouseEntered(MouseEvent me){
      if (isEnabled()){
         setForeground(Color.blue);
      }
   }
   public void mouseExited(MouseEvent me){
      setForeground(Color.black);
   }
   public void mousePressed(MouseEvent me){}
   public void mouseReleased(MouseEvent me){}
   public void mouseClicked(MouseEvent me){}
}
```

```
import java.awt.*;
public class OutlinedAutoLabel extends Panel{
   AutoLabel[] allLabels;
   OutlinedAutoLabel(String[] linesOfText, Font labelFont){
      setLayout(null);
      int maxWidth = 0;
      allLabels = new AutoLabel[linesOfText.length];
      for (int i = 0; i < allLabels.length; i++){
         allLabels[i] = new AutoLabel(linesOfText[i], labelFont);
         if (allLabels[i].getSize().width > maxWidth){
            maxWidth = allLabels[i].getSize().width;
         }
         allLabels[i].setLocation(3, 1 + (i * allLabels[0].getSize().height));
         add(allLabels[i]);
      }
      if (allLabels.length != 0){
         setSize(maxWidth +6, allLabels.length * allLabels[0].getSize().height + 2);
      }
   }
   OutlinedAutoLabel(String oneLineOfText, Font labelFont){
      setLayout(null);
      AutoLabel label = new AutoLabel(oneLineOfText, labelFont);
      label.setLocation(3, 1);
      add(label);
      setSize(label.getSize().width + 6, label.getSize().height + 2);
      allLabels = new AutoLabel[1];
      allLabels[0] = label;
   }
   public void setBackground(Color backgroundColor){
      super.setBackground(backgroundColor);
      for (int i = 0; i < allLabels.length; i++){
         allLabels[i].setBackground(backgroundColor);
      }
   }
   public void setForeground(Color foregroundColor){
      for (int i = 0; i < allLabels.length; i++){
         allLabels[i].setForeground(foregroundColor);
      }
   }
   public void paint(Graphics g){
      super.paint(g);
      g.setColor(Color.black);
      g.drawRect(0, 0, getSize().width-1, getSize().height-1);
   }

}
```

```
import java.awt.*;
import java.awt.event.*;

public class DartAppletSubmitLEAVerificationScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   MouseOverButton btnYesSubmit, btnNoGoBack;
   boolean blnFieldLeftBlank;

   public DartAppletSubmitLEAVerificationScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setLayout(null);

      btnYesSubmit = new MouseOverButton("Yes, Submit");
      btnYesSubmit.setSize(90, 20);
      btnYesSubmit.setLocation(30, 220);
      btnYesSubmit.addActionListener(this);
      btnYesSubmit.setFont(mainAppletContext.regularScreenFont);
      add(btnYesSubmit);

      btnNoGoBack = new MouseOverButton("No, Go Back");
      btnNoGoBack.setSize(90, 20);
      btnNoGoBack.setLocation(140, 220);
      btnNoGoBack.addActionListener(this);
      btnNoGoBack.setFont(mainAppletContext.regularScreenFont);
      add(btnNoGoBack);
   }

   public void setVisible(boolean visible){
      super.setVisible(visible);
      if (visible){
         blnFieldLeftBlank = false;
         btnYesSubmit.requestFocus();
      }
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.smallerScreenTitleFont);
      fm = g.getFontMetrics();

      tempString = "Submit?";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      g.setFont(mainAppletContext.regularScreenFont);
      fm = g.getFontMetrics();
      tempString = "Have you checked over all";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 105);
      tempString = "of your information?";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 135);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }

   public void btnYesSubmitClicked(){
      // check to make sure all data was filled in
      if (mainAppletContext.lrNewLEARecord.username.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtUsername);
      }
```

```
      else if (mainAppletContext.lrNewLEARecord.jurisdictionType.equals("Select One")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.chcJurisdictionType
);
      }
      else if (mainAppletContext.lrNewLEARecord.jurisdictionName.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtJurisdictionName
);
      }
      else if (mainAppletContext.lrNewLEARecord.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtRankTitle);
      }
      else if (mainAppletContext.lrNewLEARecord.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtLEAName);
      }
      else if (mainAppletContext.lrNewLEARecord.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtAddress);
      }
      else if (mainAppletContext.lrNewLEARecord.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtCity);
      }
      else if (mainAppletContext.lrNewLEARecord.state.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtContactState);
      }
      else if (mainAppletContext.lrNewLEARecord.zip.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtZip);
      }
      else if (mainAppletContext.lrNewLEARecord.voice.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtCommVoice);
      }
      else if (mainAppletContext.lrNewLEARecord.fax.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtCommFax);
      }
      else if (mainAppletContext.lrNewLEARecord.DSNVoice.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtDSNVoice);
      }
      else if (mainAppletContext.lrNewLEARecord.DSNFax.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtDSNFax);
      }
      else if (mainAppletContext.lrNewLEARecord.email.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
```

```
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtEmail);
      }
      else if (mainAppletContext.lrNewLEARecord.url.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtURLWWWIntranet);
      }
      else if (mainAppletContext.lrNewLEARecord.password.equals("")){
         blnFieldLeftBlank = true;
         mainAppletContext.scrSubmitLEAFieldLeftBlankScreen.setScreenAndComponentToJumpBackT
o(mainAppletContext.scrLEAInfoScreen, mainAppletContext.scrLEAInfoScreen.txtPassword);
      }

      if (blnFieldLeftBlank){
         mainAppletContext.showScreen(mainAppletContext.scrSubmitLEAFieldLeftBlankScreen);
      }
      else{       // send data to database
         Object[] dataTransportArray = new Object[2];
         dataTransportArray[0] = "newLEARecord";
         dataTransportArray[1] = mainAppletContext.lrNewLEARecord;
         dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
         mainAppletContext.showScreen(mainAppletContext.scrSubmitLEASuccessfulScreen);
      }
   }

   public void btnNoGoBackClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrLEAInfoScreen);
   }

   public void actionPerformed(ActionEvent ae){
      String ac = ae.getActionCommand();
      if (ac.equals("Yes, Submit")){
         btnYesSubmitClicked();
      }
      else if (ac.equals("No, Go Back")){
         btnNoGoBackClicked();
      }
   }

}
```

```
import java.awt.*;
import java.awt.event.*;

public class DartAppletSubmitLEASuccessfulScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   MouseOverButton btnOK;

   public DartAppletSubmitLEASuccessfulScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setLayout(null);

      btnOK = new MouseOverButton("OK");
      btnOK.setSize(50, 20);
      btnOK.setLocation(100, 220);
      btnOK.addActionListener(this);
      add(btnOK);
   }

   public void setVisible(boolean visible){
      super.setVisible(visible);
      if (visible){
         btnOK.requestFocus();
      }
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.smallerScreenTitleFont);
      fm = g.getFontMetrics();

      tempString = "Success";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      g.setFont(mainAppletContext.regularScreenFont);
      fm = g.getFontMetrics();
      tempString = "Information received, thank";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 105);
      tempString = "you for participating.";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 135);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }

   public void btnOKClicked(){
      mainAppletContext.showScreen(mainAppletContext.scrMainScreen);
   }

   public void actionPerformed(ActionEvent ae){
      String ac = ae.getActionCommand();
      if (ac.equals("OK")){
         btnOKClicked();
      }
   }

}
```

```
import java.awt.*;
import java.awt.event.*;

public class DartAppletSubmitLEAFieldLeftBlankScreen extends Panel implements ActionListener
{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   Panel screenToGoBackTo;
   Component componentToFocusOn;
   MouseOverButton btnOK;

   public DartAppletSubmitLEAFieldLeftBlankScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setLayout(null);

      btnOK = new MouseOverButton("OK");
      btnOK.setSize(50, 20);
      btnOK.setLocation(100, 220);
      btnOK.addActionListener(this);
      add(btnOK);
   }

   public void setVisible(boolean visible){
      super.setVisible(visible);
      if (visible){
         btnOK.requestFocus();
      }
   }

   public void setScreenAndComponentToJumpBackTo(Panel screenToGoBackTo, Component componentToFocusOn){
      this.screenToGoBackTo = screenToGoBackTo;
      this.componentToFocusOn = componentToFocusOn;
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.smallerScreenTitleFont);
      fm = g.getFontMetrics();

      tempString = "Incomplete";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 + 0.5), 40);

      g.setFont(mainAppletContext.regularScreenFont);
      fm = g.getFontMetrics();
      tempString = "You have left at least one";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 + 0.5), 105);
      tempString = "field blank.  Taking you back";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 + 0.5), 135);
      tempString = "to the first one.";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 + 0.5), 165);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }

   public void btnOKClicked(){
      mainAppletContext.showScreen(screenToGoBackTo);
      componentToFocusOn.requestFocus();
   }
```

```
    public void actionPerformed(ActionEvent ae){
        String ac = ae.getActionCommand();
        if (ac.equals("OK")){
            btnOKClicked();
        }
    }

}
```

```
import java.awt.*;
import java.awt.event.*;

public class DartAppletNewLEALoginInformationScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   Panel screenToGoBackTo;
   Component componentToFocusOn;
   MouseOverButton btnBeginITCensus;

   public DartAppletNewLEALoginInformationScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setLayout(null);

      btnBeginITCensus = new MouseOverButton("Begin  IT Census");
      btnBeginITCensus.setSize(120, 20);
      btnBeginITCensus.setLocation(90, 260);
      btnBeginITCensus.addActionListener(this);
      add(btnBeginITCensus);
   }

   public void setVisible(boolean visible){
      super.setVisible(visible);
      if (visible){
         btnBeginITCensus.requestFocus();
      }
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.smallerScreenTitleFont);
      fm = g.getFontMetrics();

      tempString = "New LEA";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      g.setFont(mainAppletContext.regularScreenFont);
      fm = g.getFontMetrics();
      tempString = "Please follow the instructions in the";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 85);
      tempString = "popup dialog box on each of the screens.";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 115);
      tempString = "The red arrow indicates the next question";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 145);
      tempString = "you should answer.";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 175);
      tempString = "Thank you for your participation.";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 225);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }

   public void btnBeginITCensusClicked(){
      setVisible(false);
      mainAppletContext.scrITCensusScreen1.setVisible(true);
   }

   public void actionPerformed(ActionEvent ae){
```

```
        String ac = ae.getActionCommand();
        if (ac.equals("Begin  IT Census")){
           btnBeginITCensusClicked();
        }
    }

}
```

```
import java.awt.*;
import java.awt.event.*;
public class DartAppletNewLEAInformationWindow extends Frame implements WindowListener, Acti
onListener, FocusListener, KeyListener{
   DartApplet mainAppletContext;
   TextArea txtMessage;
   MouseOverButton btnOK;
   DartAppletNewLEAInformationWindow(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setSize(300, 370);
      setLocation(100, 100);
      setResizable(false);
      setLayout(null);
      setTitle("Directions");
      txtMessage = new TextArea("", 0, 0, TextArea.SCROLLBARS_VERTICAL_ONLY);
      txtMessage.setFont(mainAppletContext.smallerScreenTitleFont);
      txtMessage.setEditable(false);
      txtMessage.setBackground(Color.lightGray);
      txtMessage.setForeground(Color.black);
      txtMessage.addFocusListener(this);
      add(txtMessage);
      btnOK = new MouseOverButton("OK");
      btnOK.setFont(mainAppletContext.regularScreenFont);
      btnOK.addActionListener(this);
      btnOK.addKeyListener(this);
      btnOK.setSize(40, 20);
      btnOK.setLocation(130, 250);
      add(btnOK);
      addWindowListener(this);
      addFocusListener(this);
   }

   public void show(){
      super.show();
      txtMessage.setSize(getSize().width - getInsets().left - getInsets().right, 200);
      txtMessage.setLocation(getInsets().left, getInsets().top);
   }

   public void paint(Graphics g){
      g.drawString("Either select the \"Enter\" key, \"Tab\" key or Click", 30, 300);
      g.drawString("on a question to continue.", 80, 320);
   }

   public void focusGained(FocusEvent fe){
      Object source = fe.getSource();
      if (source == txtMessage){
         txtMessage.transferFocus();
      }
   }

   public void focusLost(FocusEvent fe){}

   public void keyTyped(KeyEvent ke){}
   public void keyReleased(KeyEvent ke){}
   public void keyPressed(KeyEvent ke) {
      Object source = ke.getSource();
      if ( (source == btnOK) && (ke.getKeyCode() == KeyEvent.VK_TAB) ){
         setVisible(false);
      }
   }

   public void setMessage(String strMessage){
      txtMessage.setText(strMessage);
   }
```

```
    public void actionPerformed(ActionEvent ae){
       setVisible(false);  // right now the only action is the "OK" button being pressed
    }

    public void windowActivated(WindowEvent we){}
    public void windowClosed(WindowEvent we){}
    public void windowClosing(WindowEvent we){
       setVisible(false);
    }
    public void windowDeactivated(WindowEvent we){
       setVisible(false);
    }
    public void windowDeiconified(WindowEvent we){}
    public void windowIconified(WindowEvent we){}
    public void windowOpened(WindowEvent we){}

}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.io.*;
public class DartAppletGISTutorialScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   Vector vtrSlides;
   int currentSlideIndex = 0;
   MediaTracker mt;
   SlideAdvanceOrDecrementCanvas slideAdvancer, slideDecrementer;
   Label lblPreviousSlide, lblNextSlide;
   PictureDisplayCanvas slideDisplayer;
   DartAppletChangeScreenButton backButton, forwardButton, homeButton;
   MouseOverButton btnReturnToDART;
   RetrieveSlidesThread threadThatRetrievesTheSlides;
   boolean blnThreadThatRetrievesTheSlidesAlreadyStarted;

   TutorialImagePanel tutorialPanel;


   public DartAppletGISTutorialScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setLayout(null);

      //slides = new Image[11];
      vtrSlides = new Vector();

      lblNextSlide = new Label("Next");
      lblNextSlide.setSize(40, 20);
      lblNextSlide.setLocation(720, 190);
      add(lblNextSlide);

      slideAdvancer = new SlideAdvanceOrDecrementCanvas("advance");
      slideAdvancer.setSize(20, 80);
      slideAdvancer.setLocation(730, 220);
      add(slideAdvancer);

      lblPreviousSlide = new Label("Previous");
      lblPreviousSlide.setSize(50, 20);
      lblPreviousSlide.setLocation(10, 190);
      add(lblPreviousSlide);

      slideDecrementer = new SlideAdvanceOrDecrementCanvas("decrement");
      slideDecrementer.setSize(20, 80);
      slideDecrementer.setLocation(20, 220);
      add(slideDecrementer);

      slideDisplayer = new PictureDisplayCanvas();
      slideDisplayer.setSize(629, 472);
      slideDisplayer.setLocation(70, 60);
      add(slideDisplayer);

      btnReturnToDART = new MouseOverButton("Return To DART");
      btnReturnToDART.setSize(160, 20);
      btnReturnToDART.setLocation(300, 540);
      btnReturnToDART.addActionListener(this);
      btnReturnToDART.setFont(mainAppletContext.regularScreenFont);
      add(btnReturnToDART);

      tutorialPanel = new TutorialImagePanel();
      tutorialPanel.setSize(342, 300);
```

```
        tutorialPanel.setLocation(142, 170);
        slideDisplayer.add(tutorialPanel);

        threadThatRetrievesTheSlides = new RetrieveSlidesThread(this);
        threadThatRetrievesTheSlides.setDaemon(false);
        blnThreadThatRetrievesTheSlidesAlreadyStarted = false;

    }

    public void btnReturnToDARTClicked(){
        setVisible(false);
        mainAppletContext.scrMainScreen.setVisible(true);
    }

    public void actionPerformed(ActionEvent ae){
        String ac = ae.getActionCommand();
        if (ac.equals("Return To DART")){
            btnReturnToDARTClicked();
        }
    }

    public void displaySlide(int slideIndex){

        if ( (slideIndex < 0) || (slideIndex >= vtrSlides.size()) ){  // invalid slide
            return;
        }
        slideDisplayer.setControlSet(slideIndex);
        if ( (slideIndex == 0) ){
            slideDisplayer.add(tutorialPanel);
            WaitOnSlideRetrievalThread wosrt = new WaitOnSlideRetrievalThread(slideIndex);
            wosrt.setDaemon(false);
            wosrt.start();
        }
        else if (slideIndex == 1){
            slideDisplayer.remove(tutorialPanel);
            WaitOnSlideRetrievalThread wosrt = new WaitOnSlideRetrievalThread(slideIndex);
            wosrt.setDaemon(false);
            wosrt.start();
        }
    }

    public void setVisible(boolean visible){

        tutorialPanel.setImage(null);  // this could probably be done somewhere else...

        super.setVisible(visible);
        if (visible){
            if (! blnThreadThatRetrievesTheSlidesAlreadyStarted){
                threadThatRetrievesTheSlides.start();
                blnThreadThatRetrievesTheSlidesAlreadyStarted = true;
            }
            else{
                threadThatRetrievesTheSlides.continueRunning();
            }

            WaitOnSlideRetrievalThread wosrt = new WaitOnSlideRetrievalThread(0);
            wosrt.setDaemon(false);
            wosrt.start();

            slideDisplayer.btnStateBoundaries.blnOn = true;
            slideDisplayer.btnCountyBoundaries.blnOn = false;
            slideDisplayer.btnRoads.blnOn = false;
            slideDisplayer.btnRivers.blnOn = false;
            RetrieveGISTutorialImageThread rgtit = new RetrieveGISTutorialImageThread(this);
            rgtit.start();
```

```
      }
      else{
         if(blnThreadThatRetrievesTheSlidesAlreadyStarted){
            threadThatRetrievesTheSlides.pause();
         }
      }
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.screenTitleFont);
      fm = g.getFontMetrics();

      tempString = "GIS Tutorial Slides (" + (currentSlideIndex + 1) + " of " + vtrSlides.si
ze() + ")";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }

   public class RetrieveSlidesThread extends Thread{
      Hashtable htblRetrievedSlides;
      Component instantiator;
      int slideCount;
      boolean threadIsPaused = false;
      public RetrieveSlidesThread(Component instantiator){
         this.instantiator = instantiator;
         htblRetrievedSlides = new Hashtable();
         slideCount = 0;
      }
      public boolean isSlideRetrieved(int slideNumber){
         synchronized(htblRetrievedSlides){
            if (htblRetrievedSlides.containsKey(new Integer(slideNumber))){
               return true;
            }
            else{
               return false;
            }
         }
      }
      public void pause(){
         threadIsPaused = true;
      }
      public void continueRunning(){
         threadIsPaused = false;
      }
      public void run(){
         Image imgUnResizedImage = null;
         try{
            slideDisplayer.setWaitingForImage(true);
            mt = new MediaTracker(instantiator);
            while(true){
               if (threadIsPaused){
                  sleep(200);
               }
               else{
                  imgUnResizedImage = mainAppletContext.getImage(mainAppletContext.getDocume
ntBase(), "slides/GISTutorial/slide" + (slideCount +1) + ".jpg");
                  mt.addImage(imgUnResizedImage, 0);
                  mt.waitForAll();
                  if (mt.isErrorAny()){
                     repaint();
                     break;
                  }
```

```
                if ( (imgUnResizedImage.getWidth(null) == 627) && (imgUnResizedImage.getHe
ight(null) == 470) ){    // if already teh right size don't rescale
                   vtrSlides.addElement( imgUnResizedImage );
                }
                else{    // otherwise make it the right size
                   vtrSlides.addElement( imgUnResizedImage.getScaledInstance(627, 470, Ima
ge.SCALE_SMOOTH) );
                   mt.addImage( (Image)(vtrSlides.elementAt(slideCount)), 0);
                   mt.waitForAll();
                   imgUnResizedImage.flush();
                }

                synchronized(htblRetrievedSlides){
                   htblRetrievedSlides.put( new Integer(slideCount), " ");
                }
                slideCount++;
                repaint();
             }
          }
       }
       catch(Exception e){
          e.printStackTrace();
       }
    }
 }

 public class WaitOnSlideRetrievalThread extends Thread{
    int slideToWaitOn;
    public WaitOnSlideRetrievalThread(int slideToWaitOn){
       this.slideToWaitOn = slideToWaitOn;
    }
    public void run(){
       slideDisplayer.setWaitingForImage(true);
       while (! threadThatRetrievesTheSlides.isSlideRetrieved(slideToWaitOn) ){
          try{
             sleep(300);
          }
          catch (Exception e){
             e.printStackTrace();
          }
       }
       currentSlideIndex = slideToWaitOn;
       slideDisplayer.setImage((Image)(vtrSlides.elementAt(slideToWaitOn)));
       slideDisplayer.setWaitingForImage(false);
       repaint();
    }
 }

 public class PictureDisplayCanvas extends Panel implements ActionListener, MouseListener{
    Image pictureToDisplay;
    FontMetrics fm;
    String waitString = "Please wait, loading slide...";
    boolean blnWaitingForImageToLoad = false;
    ToggleButton btnStateBoundaries, btnCountyBoundaries, btnRoads, btnRivers;
    Button btnGISLinks1, btnGISLinks2, btnGISLinks3, btnGISLinks4;
    AutoLabel lblGISLinks1, lblGISLinks2, lblGISLinks3, lblGISLinks4, lblGISLinks4a;
    Color clrLightBlue = new Color(0, 0, 150);
    public PictureDisplayCanvas(){
       setBackground(Color.white);
       setLayout(null);

       btnStateBoundaries = new ToggleButton();
       btnStateBoundaries.setSize(25, 25);
       btnStateBoundaries.setLocation(506, 292);
       add(btnStateBoundaries);
```

```
        btnCountyBoundaries = new ToggleButton();
        btnCountyBoundaries.setSize(25, 25);
        btnCountyBoundaries.setLocation(506, 344);
        add(btnCountyBoundaries);

        btnRoads = new ToggleButton();
        btnRoads.setSize(25, 25);
        btnRoads.setLocation(506, 392);
        add(btnRoads);

        btnRivers = new ToggleButton();
        btnRivers.setSize(25, 25);
        btnRivers.setLocation(506, 435);
        add(btnRivers);

        btnGISLinks1 = new Button();
        btnGISLinks1.setSize(20, 20);
        btnGISLinks1.setLocation(100, 170);
        btnGISLinks1.addActionListener(this);

        btnGISLinks2 = new Button();
        btnGISLinks2.setSize(20, 20);
        btnGISLinks2.setLocation(100, 220);
        btnGISLinks2.addActionListener(this);

        btnGISLinks3 = new Button();
        btnGISLinks3.setSize(20, 20);
        btnGISLinks3.setLocation(100, 270);
        btnGISLinks3.addActionListener(this);

        btnGISLinks4 = new Button();
        btnGISLinks4.setSize(20, 20);
        btnGISLinks4.setLocation(100, 320);
        btnGISLinks4.addActionListener(this);

        lblGISLinks1 = new AutoLabel("What is GIS?", mainAppletContext.smallerScreenTitleFo
nt);
        lblGISLinks1.setLocation(130, 170);
        lblGISLinks1.setForeground(clrLightBlue);
        lblGISLinks1.addMouseListener(this);

        lblGISLinks2 = new AutoLabel("What kinds of data are available?", mainAppletContext
.smallerScreenTitleFont);
        lblGISLinks2.setLocation(130, 220);
        lblGISLinks2.setForeground(clrLightBlue);
        lblGISLinks2.addMouseListener(this);

        lblGISLinks3 = new AutoLabel("Who are some of the major commercial GIS providers?",
 mainAppletContext.smallerScreenTitleFont);
        lblGISLinks3.setLocation(130, 270);
        lblGISLinks3.setForeground(clrLightBlue);
        lblGISLinks3.addMouseListener(this);

        lblGISLinks4 = new AutoLabel("What's the future of GIS?  Linking Commercial and", m
ainAppletContext.smallerScreenTitleFont);
        lblGISLinks4.setLocation(130, 320);
        lblGISLinks4.setForeground(clrLightBlue);
        lblGISLinks4.addMouseListener(this);

        lblGISLinks4a = new AutoLabel("Public GIS solutions together.", mainAppletContext.s
mallerScreenTitleFont);
        lblGISLinks4a.setLocation(130, 340);
        lblGISLinks4a.setForeground(clrLightBlue);
        lblGISLinks4a.addMouseListener(this);
```

```
    }
    public void setControlSet(int controlIndex){
       if (controlIndex == 0){
          remove(btnGISLinks1);
          remove(btnGISLinks2);
          remove(btnGISLinks3);
          remove(btnGISLinks4);
          remove(lblGISLinks1);
          remove(lblGISLinks2);
          remove(lblGISLinks3);
          remove(lblGISLinks4);
          remove(lblGISLinks4a);
          add(btnStateBoundaries);
          add(btnCountyBoundaries);
          add(btnRoads);
          add(btnRivers);
       }
       else if (controlIndex == 1){
          remove(btnStateBoundaries);
          remove(btnCountyBoundaries);
          remove(btnRoads);
          remove(btnRivers);
          add(btnGISLinks1);
          add(btnGISLinks2);
          add(btnGISLinks3);
          add(btnGISLinks4);
          add(lblGISLinks1);
          add(lblGISLinks2);
          add(lblGISLinks3);
          add(lblGISLinks4);
          add(lblGISLinks4a);
       }
    }

    public void btnGISLinks1Clicked(){
       try{
          mainAppletContext.getAppletContext().showDocument( new URL("http://www.gis.com")
, "_this");
       }
       catch(Exception e){
          e.printStackTrace();
       }
    }

    public void btnGISLinks2Clicked(){
       try{
          mainAppletContext.getAppletContext().showDocument( new URL("http://www.gisdatade
pot.com"), "_this");
       }
       catch(Exception e){
          e.printStackTrace();
       }
    }

    public void btnGISLinks3Clicked(){
       try{
          mainAppletContext.getAppletContext().showDocument( new URL("http://www.gislinx.c
om/Miscellaneous/Links/index.shtml"), "_this");
       }
       catch(Exception e){
          e.printStackTrace();
       }
    }
```

```
      public void btnGISLinks4Clicked(){
         try{
            mainAppletContext.getAppletContext().showDocument( new URL("http://postoffice.nr
lssc.navy.mil/dmap/home.html"), "_this");
         }
         catch(Exception e){
            e.printStackTrace();
         }
      }

      public void actionPerformed(ActionEvent ae){
         Object eventSource = ae.getSource();
         if (eventSource == btnGISLinks1){
            btnGISLinks1Clicked();
         }
         else if (eventSource == btnGISLinks2){
            btnGISLinks2Clicked();
         }
         else if (eventSource == btnGISLinks3){
            btnGISLinks3Clicked();
         }
         else if (eventSource == btnGISLinks4){
            btnGISLinks4Clicked();
         }
      }

      public void mouseEntered(MouseEvent me){
         AutoLabel eventSource = (AutoLabel)me.getSource();
         eventSource.setForeground(Color.blue);
         if ( (eventSource == lblGISLinks4) || (eventSource == lblGISLinks4a) ){
            lblGISLinks4.setForeground(Color.blue);
            lblGISLinks4a.setForeground(Color.blue);
         }
      }
      public void mouseExited(MouseEvent me){
         AutoLabel eventSource = (AutoLabel)me.getSource();
         eventSource.setForeground(clrLightBlue);
         if ( (eventSource == lblGISLinks4) || (eventSource == lblGISLinks4a) ){
            lblGISLinks4.setForeground(clrLightBlue);
            lblGISLinks4a.setForeground(clrLightBlue);
         }
      }
      public void mousePressed(MouseEvent me){}
      public void mouseClicked(MouseEvent me){
         Object eventSource = me.getSource();
         if (eventSource == lblGISLinks1){
            btnGISLinks1Clicked();
         }
         else if (eventSource == lblGISLinks2){
            btnGISLinks2Clicked();
         }
         else if (eventSource == lblGISLinks3){
            btnGISLinks3Clicked();
         }
         else if ( (eventSource == lblGISLinks4) || (eventSource == lblGISLinks4a) ){
            btnGISLinks4Clicked();
         }
      }
      public void mouseReleased(MouseEvent me){}

      public void setImage(Image pictureToDisplay){
         this.pictureToDisplay = pictureToDisplay;
         repaint();
      }
      public void setWaitingForImage(boolean waiting){
```

```
        blnWaitingForImageToLoad = waiting;
        repaint();
      }
      public void paint(Graphics g){
         g.setFont(mainAppletContext.smallerScreenTitleFont);
         if (fm == null){
            fm = g.getFontMetrics();
         }
         g.setColor(Color.black);
         if (blnWaitingForImageToLoad){
            g.drawString(waitString, (int)(getSize().width/2.0 - fm.stringWidth(waitString)/
2.0 + 0.5), 100);
         }
         else{
            if (pictureToDisplay != null){
               g.drawImage(pictureToDisplay, 1, 1, null);
            }
         }
         g.drawRect(0, 0, getSize().width -1, getSize().height -1);
      }
      public void update(Graphics g){
         if (blnWaitingForImageToLoad){
            g.setColor(getBackground());
            g.fillRect(0, 0, getSize().width, getSize().height);
            paint(g);
         }
         else{
            paint(g);
         }
      }
   }

   public class SlideAdvanceOrDecrementCanvas extends Canvas implements MouseListener, Mouse
MotionListener{
      boolean blnMouseInside = false;
      boolean blnMouseInsidePreviously = false;
      Polygon plgSymbol;
      String direction;
      Image buffer;
      Graphics bufferG;
      public SlideAdvanceOrDecrementCanvas(String direction){
         this.direction = direction;
         addMouseListener(this);
         addMouseMotionListener(this);
      }
      public void paint(Graphics g){
         if (buffer == null){
            buffer = createImage(getSize().width, getSize().height);
            bufferG = buffer.getGraphics();
         }
         if (plgSymbol == null){
            plgSymbol = new Polygon();
            if (direction.equals("advance")){
               plgSymbol.addPoint(0, 0);
               plgSymbol.addPoint(0, getSize().height -1);
               plgSymbol.addPoint(getSize().width -1, (int)(getSize().height/2.0 + 0.5));
            }
            else if (direction.equals("decrement")){
               plgSymbol.addPoint(0, (int)(getSize().height/2.0 + 0.5));
               plgSymbol.addPoint(getSize().width -1, 0);
               plgSymbol.addPoint(getSize().width -1, getSize().height -1);
            }
         }
         if (blnMouseInside){
            bufferG.setColor(Color.blue);
```

```
        }
        else{
           bufferG.setColor(Color.black);
        }
        bufferG.fillPolygon(plgSymbol);
        g.drawImage(buffer, 0, 0, null);
     }
     public void update(Graphics g){
        paint(g);
     }
     public void mousePressed(MouseEvent me){}
     public void mouseReleased(MouseEvent me){}
     public void mouseClicked(MouseEvent me){
        if (blnMouseInside){
           if (direction.equals("advance")){
              displaySlide(currentSlideIndex + 1);
           }
           else if (direction.equals("decrement")){
              displaySlide(currentSlideIndex - 1);
           }
        }
     }
     public void mouseEntered(MouseEvent me){}
     public void mouseExited(MouseEvent me){
        blnMouseInside = false;
        repaint();
     }
     public void mouseMoved(MouseEvent me){
       blnMouseInsidePreviously = blnMouseInside;
       if (plgSymbol.contains(me.getX(), me.getY())){
          blnMouseInside = true;
       }
       else{
          blnMouseInside = false;
       }
       if ( blnMouseInsidePreviously != blnMouseInside){
          repaint();
       }
     }
     public void mouseDragged(MouseEvent me){}
  }

  class TutorialImagePanel extends Panel{
     Image tutorialImage = null;
     public void setImage(Image imageToDisplay){
        if (tutorialImage != null){
           tutorialImage.flush();
        }
        tutorialImage = imageToDisplay;
        repaint();
     }
     public void paint(Graphics g){
        if (tutorialImage != null){
           g.drawImage(tutorialImage, 0, 0, null);
        }
        g.drawRect(0, 0, getSize().width -1, getSize().height -1);
     }
  }


  class ToggleButton extends Canvas implements MouseListener{
     boolean blnOn = false;
     int width, height, widthMinus1, widthMinus2, heightMinus1, heightMinus2;
     ToggleButton(){
        addMouseListener(this);
```

```
        setBackground(Color.lightGray);
    }
    public void mouseEntered(MouseEvent me){}
    public void mouseExited(MouseEvent me){}
    public void mousePressed(MouseEvent me){}
    public void mouseClicked(MouseEvent me){
        blnOn = !blnOn;
        repaint();
        RetrieveGISTutorialImageThread rgtit = new RetrieveGISTutorialImageThread(this);
        rgtit.start();
    }
    public void mouseReleased(MouseEvent me){}
    public void paint(Graphics g){
        width = getSize().width;
        height = getSize().height;
        widthMinus1 = width -1;
        widthMinus2 = width -2;
        heightMinus1 = height -1;
        heightMinus2 = height -2;
        if (blnOn){    // button pushed down
            g.setColor(Color.white);
            g.drawLine(0, heightMinus1, widthMinus1, heightMinus1);
            g.drawLine(0, heightMinus2, widthMinus1, heightMinus2);
            g.drawLine(widthMinus1, 0, widthMinus1, heightMinus1);
            g.drawLine(widthMinus2, 0, widthMinus2, heightMinus1);
            g.setColor(Color.black);
            g.drawLine(0, 0, widthMinus1, 0);
            g.setColor(Color.gray);
            g.drawLine(0, 1, widthMinus2, 1);
            g.setColor(Color.black);
            g.drawLine(0, 0, 0, heightMinus1);
            g.setColor(Color.gray);
            g.drawLine(1, 0, 1, heightMinus2);

            g.setColor(Color.lightGray);
            g.drawLine(widthMinus1, 0, widthMinus1, heightMinus1);
            g.drawLine(0, heightMinus1, widthMinus1, heightMinus1);
        }
        else{    // button not pushed down
            g.setColor(Color.white);
            g.drawLine(0, 0, widthMinus1, 0);
            g.drawLine(0, 1, widthMinus1, 1);
            g.drawLine(0, 0, 0, heightMinus1);
            g.drawLine(1, 0, 1, heightMinus1);
            g.setColor(Color.black);
            g.drawLine(0, heightMinus1, widthMinus1, heightMinus1);
            g.setColor(Color.gray);
            g.drawLine(1, heightMinus2, widthMinus1, heightMinus2);
            g.setColor(Color.black);
            g.drawLine(widthMinus1, 0, widthMinus1, heightMinus1);
            g.setColor(Color.gray);
            g.drawLine(widthMinus2, 1, widthMinus2, heightMinus1);

            g.setColor(Color.lightGray);
            g.drawLine(0, 0, widthMinus1, 0);
            g.drawLine(0, 0, 0, heightMinus1);
        }
    }
}

class RetrieveGISTutorialImageThread extends Thread{
    Component componentForMediaTracker;
    Image imgGISTutorialImage;
    RetrieveGISTutorialImageThread(Component componentForMediaTracker){
        this.componentForMediaTracker = componentForMediaTracker;
```

```
    }
    public void run(){
       try{
          if (imgGISTutorialImage != null){
             imgGISTutorialImage.flush();
             imgGISTutorialImage = null;
          }

          Object[] dataTransportArray = null;

          String strBitString = "";
          if (slideDisplayer.btnStateBoundaries.blnOn){
             strBitString = strBitString + "1";
          }
          else{
             strBitString = strBitString + "0";
          }
          if (slideDisplayer.btnCountyBoundaries.blnOn){
             strBitString = strBitString + "1";
          }
          else{
             strBitString = strBitString + "0";
          }
          if (slideDisplayer.btnRoads.blnOn){
             strBitString = strBitString + "1";
          }
          else{
             strBitString = strBitString + "0";
          }
          if (slideDisplayer.btnRivers.blnOn){
             strBitString = strBitString + "1";
          }
          else{
             strBitString = strBitString + "0";
          }
          dataTransportArray = new Object[2];
          dataTransportArray[0] = "GISTutorialImageRequest";
          dataTransportArray[1] = strBitString;

          dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);

          String filePrefix = (String)dataTransportArray[0];
          String imageFileName = filePrefix + ".jpg";
          String fileFinishedMarkerFileName = filePrefix + ".finished";
          URL finishedFileURL = new URL(mainAppletContext.getCodeBase() + fileFinishedMark
erFileName);
          boolean fileCreated = false;
          int numberOfTries = 0;
          InputStream tempIS = null;
          while ( (! fileCreated) && (numberOfTries < 100) ){
             try{
                sleep(200);
                tempIS = finishedFileURL.openStream();
                tempIS.close();
                fileCreated = true;
             }
             catch(Exception fileNotYetThereException){
                numberOfTries++;
                if (numberOfTries == 100){
                   System.out.println("GIS Tutorial Image File Not Found");
                }
             }
          }
          if (fileCreated){
             Image imgUnscaledGISTutorialImage = mainAppletContext.getImage(mainAppletCont
```

```
ext.getCodeBase(), imageFileName);
                MediaTracker mt = new MediaTracker(componentForMediaTracker);
                mt.addImage(imgUnscaledGISTutorialImage, 0);
                mt.waitForAll();
                mt.removeImage(imgUnscaledGISTutorialImage, 0);
                imgGISTutorialImage = imgUnscaledGISTutorialImage.getScaledInstance( 342, 300
, Image.SCALE_SMOOTH);
                mt.addImage(imgGISTutorialImage, 0);
                mt.waitForAll();
                imgUnscaledGISTutorialImage.flush();
                tutorialPanel.setImage(imgGISTutorialImage);
                dataTransportArray = new Object[2];
                dataTransportArray[0] = "deleteImage";
                dataTransportArray[1] = imageFileName.substring(imageFileName.indexOf("/"), i
mageFileName.lastIndexOf("."));
                dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
            }
         }
         catch(Exception e){
            e.printStackTrace();
         }
      }
   }

}
```

```
import java.awt.*;
import java.awt.event.*;

public class DartAppletLEAITCensusWindow extends Frame implements KeyListener, ActionListene
r, WindowListener, ItemListener{
   DartApplet mainAppletContext;
   ScrollPane questionsScrollPane;
   Panel questionsPanel;
   Panel scrollPaneContainerPanel;

   OutlinedAutoLabel lblQuestion1;
   OutlinedAutoLabel lblQuestion2;
   OutlinedAutoLabel lblQuestion3;

   Checkbox chkAccessYes, chkAccessNo;
   CheckboxGroup cbgAccess;

   Choice chcTypeOfComputer, chcSpeedOfComputer;

   OutlinedAutoLabel lblQuestion4;

   Choice chcOperatingSystem;

   OutlinedAutoLabel lblQuestion5, lblQuestion6, lblQuestion7;

   Checkbox chkNetworkConnectionYes, chkNetworkConnectionNo;
   CheckboxGroup cbgNetworkConnection;

   Choice chcTypeOfNetworkConnection, chcSpeedOfNetworkConnection;

   OutlinedAutoLabel lblQuestion8, lblQuestion9, lblQuestion10;

   Checkbox chkBrowserYes, chkBrowserNo;
   CheckboxGroup cbgBrowser;

   Choice chcBrowserName;

   Checkbox chkViewedWebPageYes, chkViewedWebPageNo;
   CheckboxGroup cbgViewedWebPage;

   MouseOverButton btnSubmit;

   LEARecord currentRecord;

   DartAppletLEAITCensusWindow(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);
      setSize(700, 730);
      setLocation(100, 10);
      setResizable(false);
      setTitle("IT Census");
      addWindowListener(this);

      addNotify();


      scrollPaneContainerPanel = new Panel();
      scrollPaneContainerPanel.setSize(getSize().width - getInsets().right - getInsets().lef
t, getSize().height - getInsets().top - getInsets().bottom + 5);
      scrollPaneContainerPanel.setLocation(getInsets().left, getInsets().top-5);
      scrollPaneContainerPanel.setLayout( new FlowLayout() );
      add(scrollPaneContainerPanel);
```

```
        questionsScrollPane = new ScrollPane(ScrollPane.SCROLLBARS_AS_NEEDED);
        questionsScrollPane.setSize(scrollPaneContainerPanel.getSize().width, scrollPaneContai
nerPanel.getSize().height-5);
        questionsScrollPane.setLocation(0, 0);

        Adjustable vadjust = questionsScrollPane.getVAdjustable();
        vadjust.setUnitIncrement(15);

        questionsPanel = new Panel();
        questionsPanel.setLayout(null);
        questionsPanel.setSize(questionsScrollPane.getSize().width - 25, 660);
        questionsPanel.setLocation(0, 0);

        lblQuestion1 = new OutlinedAutoLabel("1. Do you have regular access to a computer to s
upport your LEA duties?", mainAppletContext.smallestScreenTitleFont);
        lblQuestion1.setLocation(50, 20);
        lblQuestion1.setBackground(Color.white);
        lblQuestion1.setForeground(Color.blue);
        questionsPanel.add(lblQuestion1);

        cbgAccess = new CheckboxGroup();

        chkAccessYes = new Checkbox("Yes", false, cbgAccess);
        chkAccessYes.setSize(60, 20);
        chkAccessYes.setLocation(250, 50);
        chkAccessYes.setFont(mainAppletContext.regularScreenFont);
        chkAccessYes.addItemListener(this);
        questionsPanel.add(chkAccessYes);

        chkAccessNo = new Checkbox("No", false, cbgAccess);
        chkAccessNo.setSize(60, 20);
        chkAccessNo.setLocation(330, 50);
        chkAccessNo.setFont(mainAppletContext.regularScreenFont);
        chkAccessNo.addItemListener(this);
        questionsPanel.add(chkAccessNo);

        lblQuestion2 = new OutlinedAutoLabel("2. Select the type of computer you have access t
o during your LEA duties.", mainAppletContext.smallestScreenTitleFont);
        lblQuestion2.setLocation(50, 80);
        lblQuestion2.setBackground(Color.white);
        lblQuestion2.setForeground(Color.blue);
        questionsPanel.add(lblQuestion2);

        chcTypeOfComputer = new Choice();
        chcTypeOfComputer.setSize(300, 20);
        chcTypeOfComputer.setLocation(165, 110);
        chcTypeOfComputer.setFont(mainAppletContext.regularScreenFont);
        chcTypeOfComputer.add("Select One");
        chcTypeOfComputer.add("Common Desktop Personal Computer (PC)");
        chcTypeOfComputer.add("Macintosh");
        chcTypeOfComputer.add("High End Workstation");
        chcTypeOfComputer.add("Internet PC Terminal");
        chcTypeOfComputer.add("Other");
        chcTypeOfComputer.add("None");
        chcTypeOfComputer.addItemListener(this);
        questionsPanel.add(chcTypeOfComputer);

        lblQuestion3 = new OutlinedAutoLabel("3. What is the functional level or speed of the
computer used in your LEA duties?", mainAppletContext.smallestScreenTitleFont);
        lblQuestion3.setLocation(50, 140);
        lblQuestion3.setBackground(Color.white);
        lblQuestion3.setForeground(Color.blue);
        questionsPanel.add(lblQuestion3);

        chcSpeedOfComputer = new Choice();
```

```
        chcSpeedOfComputer.setSize(300, 20);
        chcSpeedOfComputer.setLocation(165, 170);
        chcSpeedOfComputer.setFont(mainAppletContext.regularScreenFont);
        chcSpeedOfComputer.add("Select One");
        chcSpeedOfComputer.add("< 200Mhz");
        chcSpeedOfComputer.add(">= 200Mhz");
        chcSpeedOfComputer.add("Other");
        chcSpeedOfComputer.add("None");
        chcSpeedOfComputer.addItemListener(this);
        questionsPanel.add(chcSpeedOfComputer);

        lblQuestion4 = new OutlinedAutoLabel("4. What kind of operating system does your compu
ter use?", mainAppletContext.smallestScreenTitleFont);
        lblQuestion4.setLocation(50, 200);
        lblQuestion4.setBackground(Color.white);
        lblQuestion4.setForeground(Color.blue);
        questionsPanel.add(lblQuestion4);

        chcOperatingSystem = new Choice();
        chcOperatingSystem.setSize(300, 20);
        chcOperatingSystem.setLocation(165, 230);
        chcOperatingSystem.setFont(mainAppletContext.regularScreenFont);
        chcOperatingSystem.add("Select One");
        chcOperatingSystem.add("MS Windows 95, 98, or 2000");
        chcOperatingSystem.add("Windows NT");
        chcOperatingSystem.add("Macintosh OS");
        chcOperatingSystem.add("UNIX");
        chcOperatingSystem.add("Other");
        chcOperatingSystem.add("None");
        chcOperatingSystem.addItemListener(this);
        questionsPanel.add(chcOperatingSystem);

        lblQuestion5 = new OutlinedAutoLabel("5. Is this computer connected to a network?", ma
inAppletContext.smallestScreenTitleFont);
        lblQuestion5.setLocation(50, 270);
        lblQuestion5.setBackground(Color.white);
        lblQuestion5.setForeground(Color.blue);
        questionsPanel.add(lblQuestion5);

        cbgNetworkConnection = new CheckboxGroup();

        chkNetworkConnectionYes = new Checkbox("Yes", false, cbgNetworkConnection);
        chkNetworkConnectionYes.setSize(60, 20);
        chkNetworkConnectionYes.setLocation(250, 300);
        chkNetworkConnectionYes.setFont(mainAppletContext.regularScreenFont);
        chkNetworkConnectionYes.addItemListener(this);
        questionsPanel.add(chkNetworkConnectionYes);

        chkNetworkConnectionNo = new Checkbox("No", false, cbgNetworkConnection);
        chkNetworkConnectionNo.setSize(60, 20);
        chkNetworkConnectionNo.setLocation(330, 300);
        chkNetworkConnectionNo.setFont(mainAppletContext.regularScreenFont);
        chkNetworkConnectionNo.addItemListener(this);
        questionsPanel.add(chkNetworkConnectionNo);

        lblQuestion6 = new OutlinedAutoLabel("6. What kind of network connection do you have?"
, mainAppletContext.smallestScreenTitleFont);
        lblQuestion6.setLocation(50, 330);
        lblQuestion6.setBackground(Color.white);
        lblQuestion6.setForeground(Color.blue);
        questionsPanel.add(lblQuestion6);

        chcTypeOfNetworkConnection = new Choice();
        chcTypeOfNetworkConnection.setSize(300, 20);
        chcTypeOfNetworkConnection.setLocation(165, 360);
```

```
        chcTypeOfNetworkConnection.setFont(mainAppletContext.regularScreenFont);
        chcTypeOfNetworkConnection.add("Select One");
        chcTypeOfNetworkConnection.add("Modem over Phone Line");
        chcTypeOfNetworkConnection.add("Local Router");
        chcTypeOfNetworkConnection.add("High Speed Multiplexer");
        chcTypeOfNetworkConnection.add("Unknown");
        chcTypeOfNetworkConnection.add("Other");
        chcTypeOfNetworkConnection.add("None");
        chcTypeOfNetworkConnection.addItemListener(this);
        questionsPanel.add(chcTypeOfNetworkConnection);

        lblQuestion7 = new OutlinedAutoLabel("7. What speed is your network connection?", main
AppletContext.smallestScreenTitleFont);
        lblQuestion7.setLocation(50, 390);
        lblQuestion7.setBackground(Color.white);
        lblQuestion7.setForeground(Color.blue);
        questionsPanel.add(lblQuestion7);

        chcSpeedOfNetworkConnection = new Choice();
        chcSpeedOfNetworkConnection.setSize(300, 20);
        chcSpeedOfNetworkConnection.setLocation(165, 420);
        chcSpeedOfNetworkConnection.setFont(mainAppletContext.regularScreenFont);
        chcSpeedOfNetworkConnection.add("Select One");
        chcSpeedOfNetworkConnection.add("56K or greater");
        chcSpeedOfNetworkConnection.add("T1 or greater");
        chcSpeedOfNetworkConnection.add("Unknown");
        chcSpeedOfNetworkConnection.add("Other");
        chcSpeedOfNetworkConnection.add("None");
        chcSpeedOfNetworkConnection.addItemListener(this);
        questionsPanel.add(chcSpeedOfNetworkConnection);

        lblQuestion8 = new OutlinedAutoLabel("8. Does this computer have browser software to c
onnect to the world wide web?", mainAppletContext.smallestScreenTitleFont);
        lblQuestion8.setLocation(50, 450);
        lblQuestion8.setBackground(Color.white);
        lblQuestion8.setForeground(Color.blue);
        questionsPanel.add(lblQuestion8);

        cbgBrowser = new CheckboxGroup();

        chkBrowserYes = new Checkbox("Yes", false, cbgBrowser);
        chkBrowserYes.setSize(60, 20);
        chkBrowserYes.setLocation(250, 480);
        chkBrowserYes.setFont(mainAppletContext.regularScreenFont);
        chkBrowserYes.addItemListener(this);
        questionsPanel.add(chkBrowserYes);

        chkBrowserNo = new Checkbox("No", false, cbgBrowser);
        chkBrowserNo.setSize(60, 20);
        chkBrowserNo.setLocation(330, 480);
        chkBrowserNo.setFont(mainAppletContext.regularScreenFont);
        chkBrowserNo.addItemListener(this);
        questionsPanel.add(chkBrowserNo);

        lblQuestion9 = new OutlinedAutoLabel("9. What kind of browser software do you have?",
mainAppletContext.smallestScreenTitleFont);
        lblQuestion9.setLocation(50, 510);
        lblQuestion9.setBackground(Color.white);
        lblQuestion9.setForeground(Color.blue);
        questionsPanel.add(lblQuestion9);

        chcBrowserName = new Choice();
        chcBrowserName.setSize(300, 20);
        chcBrowserName.setLocation(165, 540);
        chcBrowserName.setFont(mainAppletContext.regularScreenFont);
```

```
        chcBrowserName.add("Select One");
        chcBrowserName.add("Netscape Navigator");
        chcBrowserName.add("MS Internet Explorer");
        chcBrowserName.add("Unknown");
        chcBrowserName.add("Other");
        chcBrowserName.add("None");
        chcBrowserName.addItemListener(this);
        questionsPanel.add(chcBrowserName);

        lblQuestion10 = new OutlinedAutoLabel("10. Have you ever viewed a webpage using this c
omputer?", mainAppletContext.smallestScreenTitleFont);
        lblQuestion10.setLocation(50, 570);
        lblQuestion10.setBackground(Color.white);
        lblQuestion10.setForeground(Color.blue);
        questionsPanel.add(lblQuestion10);

        cbgViewedWebPage = new CheckboxGroup();

        chkViewedWebPageYes = new Checkbox("Yes", false, cbgViewedWebPage);
        chkViewedWebPageYes.setSize(60, 20);
        chkViewedWebPageYes.setLocation(250, 600);
        chkViewedWebPageYes.setFont(mainAppletContext.regularScreenFont);
        chkViewedWebPageYes.addItemListener(this);
        questionsPanel.add(chkViewedWebPageYes);

        chkViewedWebPageNo = new Checkbox("No", false, cbgViewedWebPage);
        chkViewedWebPageNo.setSize(60, 20);
        chkViewedWebPageNo.setLocation(330, 600);
        chkViewedWebPageNo.setFont(mainAppletContext.regularScreenFont);
        chkViewedWebPageNo.addItemListener(this);
        questionsPanel.add(chkViewedWebPageNo);

        btnSubmit = new MouseOverButton("Submit");
        btnSubmit.setSize(80, 20);
        btnSubmit.setLocation(260, 640);
        btnSubmit.addActionListener(this);
        btnSubmit.setFont(mainAppletContext.regularScreenFont);
        btnSubmit.setVisible(false);
        questionsPanel.add(btnSubmit);

        questionsScrollPane.add(questionsPanel, FlowLayout.LEFT);

        scrollPaneContainerPanel.add(questionsScrollPane);

        pack();
    }

    public void clearFields(){
        chkAccessYes.setCheckboxGroup(null);
        chkAccessNo.setCheckboxGroup(null);
        chkAccessYes.setState(false);
        chkAccessNo.setState(false);
        chkAccessYes.setCheckboxGroup(cbgAccess);
        chkAccessNo.setCheckboxGroup(cbgAccess);
        chcTypeOfComputer.select(0);
        chcSpeedOfComputer.select(0);
        chcOperatingSystem.select(0);
        chkNetworkConnectionYes.setCheckboxGroup(null);
        chkNetworkConnectionNo.setCheckboxGroup(null);
        chkNetworkConnectionYes.setState(false);
        chkNetworkConnectionNo.setState(false);
        chkNetworkConnectionYes.setCheckboxGroup(cbgNetworkConnection);
        chkNetworkConnectionNo.setCheckboxGroup(cbgNetworkConnection);
        chcTypeOfNetworkConnection.select(0);
        chcSpeedOfNetworkConnection.select(0);
```

```
        chkBrowserYes.setCheckboxGroup(null);
        chkBrowserNo.setCheckboxGroup(null);
        chkBrowserYes.setState(false);
        chkBrowserNo.setState(false);
        chkBrowserYes.setCheckboxGroup(cbgBrowser);
        chkBrowserNo.setCheckboxGroup(cbgBrowser);
        chcBrowserName.select(0);
        chkViewedWebPageYes.setCheckboxGroup(null);
        chkViewedWebPageNo.setCheckboxGroup(null);
        chkViewedWebPageYes.setState(false);
        chkViewedWebPageNo.setState(false);
        chkViewedWebPageYes.setCheckboxGroup(cbgViewedWebPage);
        chkViewedWebPageNo.setCheckboxGroup(cbgViewedWebPage);
        currentRecord = null;
    }

    public void setFields(){
        if (currentRecord.strAccessToComputer.equals("Yes")){
            cbgAccess.setSelectedCheckbox(chkAccessYes);
        }
        else if (currentRecord.strAccessToComputer.equals("No")){
            cbgAccess.setSelectedCheckbox(chkAccessNo);
        }
        chcTypeOfComputer.select(currentRecord.strTypeOfComputer);
        chcSpeedOfComputer.select(currentRecord.strSpeedOfComputer);
        chcOperatingSystem.select(currentRecord.strOperatingSystem);
        if (currentRecord.strConnectedToNetwork.equals("Yes")){
            cbgNetworkConnection.setSelectedCheckbox(chkNetworkConnectionYes);
        }
        else if (currentRecord.strConnectedToNetwork.equals("No")){
            cbgNetworkConnection.setSelectedCheckbox(chkNetworkConnectionNo);
        }
        chcTypeOfNetworkConnection.select(currentRecord.strTypeOfNetworkConnection);
        chcSpeedOfNetworkConnection.select(currentRecord.strSpeedOfNetworkConnection);
        if (currentRecord.strBrowser.equals("Yes")){
            cbgBrowser.setSelectedCheckbox(chkBrowserYes);
        }
        else if (currentRecord.strBrowser.equals("No")){
            cbgBrowser.setSelectedCheckbox(chkBrowserNo);
        }
        chcBrowserName.select(currentRecord.strTypeOfBrowser);
        if (currentRecord.strViewedWebPage.equals("Yes")){
            cbgViewedWebPage.setSelectedCheckbox(chkViewedWebPageYes);
        }
        else if (currentRecord.strViewedWebPage.equals("No")){
            cbgViewedWebPage.setSelectedCheckbox(chkViewedWebPageNo);
        }
    }

    public void setEnabledState(boolean enabledState){
        chkAccessYes.setEnabled(enabledState);
        chkAccessNo.setEnabled(enabledState);
        chcTypeOfComputer.setEnabled(enabledState);
        chcSpeedOfComputer.setEnabled(enabledState);
        chcOperatingSystem.setEnabled(enabledState);
        chkNetworkConnectionYes.setEnabled(enabledState);
        chkNetworkConnectionNo.setEnabled(enabledState);
        chcTypeOfNetworkConnection.setEnabled(enabledState);
        chcSpeedOfNetworkConnection.setEnabled(enabledState);
        chkBrowserYes.setEnabled(enabledState);
        chkBrowserNo.setEnabled(enabledState);
        chcBrowserName.setEnabled(enabledState);
        chkViewedWebPageYes.setEnabled(enabledState);
        chkViewedWebPageNo.setEnabled(enabledState);
    }
```

```
   public void setState1(){          // this state is where a LEA or CDC has logged in
      clearFields();
      setEnabledState(false);
      if (mainAppletContext.fpNextItemPrompt.getParent() != null){
         mainAppletContext.fpNextItemPrompt.getParent().remove(mainAppletContext.fpNextItemP
rompt);
      }
      currentRecord = mainAppletContext.scrLEAInfoScreen.currentRecord;
      setFields();
      btnSubmit.setVisible(false);
   }

   public void setState2(){              // this state is where a new LEA is entering infor
mation and has not gone through the 'script'
      clearFields();
      setEnabledState(false);
      if (mainAppletContext.fpNextItemPrompt.getParent() != null){
         mainAppletContext.fpNextItemPrompt.getParent().remove(mainAppletContext.fpNextItemP
rompt);
      }
      mainAppletContext.fpNextItemPrompt.setDirection("right");
      questionsPanel.add(mainAppletContext.fpNextItemPrompt);
      mainAppletContext.fpNextItemPrompt.setLocation(chkAccessYes.getLocation().x - 30, chkA
ccessYes.getLocation().y);
      currentRecord = mainAppletContext.scrLEAInfoScreen.currentRecord;
      setFields();
      chkAccessYes.setEnabled(true);
      chkAccessNo.setEnabled(true);
      btnSubmit.setEnabled(false);
      btnSubmit.setVisible(true);
      mainAppletContext.wndNewLEAInformationWindow.setMessage("Please answer these questions
 regarding the type of computer you have, the operating system you are using, and your netwo
rk setup.");
      mainAppletContext.wndNewLEAInformationWindow.show();
   }

   public void setState3(){      // this state is where a new LEA is entering information an
d has gone through the 'script'
      clearFields();
      setEnabledState(true);
      if (mainAppletContext.fpNextItemPrompt.getParent() != null){
         mainAppletContext.fpNextItemPrompt.getParent().remove(mainAppletContext.fpNextItemP
rompt);
      }
      currentRecord = mainAppletContext.scrLEAInfoScreen.currentRecord;
      setFields();
      btnSubmit.setEnabled(true);
      btnSubmit.setVisible(true);
   }

   public void show(/*boolean visible*/){

      if (! mainAppletContext.blnInNewLEAMode){
         setState1();
      }
      else if (! mainAppletContext.blnNewLEAHasGoneThroughScript){
         setState2();
      }
      else{
         setState3();
      }

      super.show();
```

```
    }

    public void btnSubmitClicked(){
        mainAppletContext.blnNewLEAHasGoneThroughScript = true;
        currentRecord.strAccessToComputer = cbgAccess.getSelectedCheckbox().getLabel();
        currentRecord.strTypeOfComputer = chcTypeOfComputer.getSelectedItem();
        currentRecord.strSpeedOfComputer = chcSpeedOfComputer.getSelectedItem();
        currentRecord.strOperatingSystem = chcOperatingSystem.getSelectedItem();
        currentRecord.strConnectedToNetwork = cbgNetworkConnection.getSelectedCheckbox().getLa
bel();
        currentRecord.strTypeOfNetworkConnection = chcTypeOfNetworkConnection.getSelectedItem(
);
        currentRecord.strSpeedOfNetworkConnection = chcSpeedOfNetworkConnection.getSelectedIte
m();
        currentRecord.strBrowser = cbgBrowser.getSelectedCheckbox().getLabel();
        currentRecord.strTypeOfBrowser = chcBrowserName.getSelectedItem();
        currentRecord.strViewedWebPage = cbgViewedWebPage.getSelectedCheckbox().getLabel();
        mainAppletContext.showScreen(mainAppletContext.scrSubmitLEAVerificationScreen);
    }

    public void actionPerformed(ActionEvent ae){
        Object eventSource= ae.getSource();
        if (eventSource == btnSubmit){
            btnSubmitClicked();
        }
    }

    public void setRemainingFieldsToNone(){
        chcTypeOfComputer.select("None");
        chcSpeedOfComputer.select("None");
        chcOperatingSystem.select("None");
        cbgNetworkConnection.setSelectedCheckbox(chkNetworkConnectionNo);
        chcTypeOfNetworkConnection.select("None");
        chcSpeedOfNetworkConnection.select("None");
        cbgBrowser.setSelectedCheckbox(chkBrowserNo);
        chcBrowserName.select("None");
        cbgViewedWebPage.setSelectedCheckbox(chkViewedWebPageNo);
    }

    public void itemStateChanged(ItemEvent ie){
        Object eventSource = ie.getSource();
        if ( (eventSource == chkAccessYes) || (eventSource == chkAccessNo) ){
            if (eventSource == chkAccessNo){
                setRemainingFieldsToNone();
                mainAppletContext.fpNextItemPrompt.setDirection("right");
                mainAppletContext.fpNextItemPrompt.setLocation(btnSubmit.getLocation().x -30, bt
nSubmit.getLocation().y);
                btnSubmit.setEnabled(true);
                mainAppletContext.wndNewLEAInformationWindow.setMessage("You have indicated that
 you do not have any computer resources.  Since this is the case, you may go ahead and press
 the \"Submit\" button without selecting the remaining fields.  If you want to select the re
maining fields, you will first have to change the first answer to \"Yes\".");
                mainAppletContext.wndNewLEAInformationWindow.show();
            }
            else{
                mainAppletContext.fpNextItemPrompt.setDirection("right");
                mainAppletContext.fpNextItemPrompt.setLocation(chcTypeOfComputer.getLocation().x
 - 30, chcTypeOfComputer.getLocation().y);
                chcTypeOfComputer.setEnabled(true);
            }
        }
        else if (eventSource == chcTypeOfComputer){
            if (! chcTypeOfComputer.getSelectedItem().equals("Select One")){    // "Select One"
is not a valid selection
                mainAppletContext.fpNextItemPrompt.setDirection("right");
```

```
            mainAppletContext.fpNextItemPrompt.setLocation(chcSpeedOfComputer.getLocation().
x - 30, chcSpeedOfComputer.getLocation().y);
            chcSpeedOfComputer.setEnabled(true);
         }
      }
      else if (eventSource == chcSpeedOfComputer){
         if (! chcSpeedOfComputer.getSelectedItem().equals("Select One")){
            mainAppletContext.fpNextItemPrompt.setDirection("right");
            mainAppletContext.fpNextItemPrompt.setLocation(chcOperatingSystem.getLocation().
x - 30, chcOperatingSystem.getLocation().y);
            chcOperatingSystem.setEnabled(true);
         }
      }
      else if (eventSource == chcOperatingSystem){
         if (! chcOperatingSystem.getSelectedItem().equals("Select One")){
            mainAppletContext.fpNextItemPrompt.setDirection("right");
            mainAppletContext.fpNextItemPrompt.setLocation(chkNetworkConnectionYes.getLocati
on().x - 30, chkNetworkConnectionYes.getLocation().y);
            chkNetworkConnectionYes.setEnabled(true);
            chkNetworkConnectionNo.setEnabled(true);
         }
      }
      else if ( (eventSource == chkNetworkConnectionYes) || (eventSource == chkNetworkConnec
tionNo) ){
         mainAppletContext.fpNextItemPrompt.setDirection("right");
         mainAppletContext.fpNextItemPrompt.setLocation(chcTypeOfNetworkConnection.getLocati
on().x - 30, chcTypeOfNetworkConnection.getLocation().y);
         chcTypeOfNetworkConnection.setEnabled(true);
      }
      else if (eventSource == chcTypeOfNetworkConnection){
         if (! chcTypeOfNetworkConnection.getSelectedItem().equals("Select One")){
            mainAppletContext.fpNextItemPrompt.setDirection("right");
            mainAppletContext.fpNextItemPrompt.setLocation(chcSpeedOfNetworkConnection.getLo
cation().x - 30, chcSpeedOfNetworkConnection.getLocation().y);
            chcSpeedOfNetworkConnection.setEnabled(true);
         }
      }
      else if (eventSource == chcSpeedOfNetworkConnection){
         if (! chcSpeedOfNetworkConnection.getSelectedItem().equals("Select One")){
            mainAppletContext.fpNextItemPrompt.setDirection("right");
            mainAppletContext.fpNextItemPrompt.setLocation(chkBrowserYes.getLocation().x - 3
0, chkBrowserYes.getLocation().y);
            chkBrowserYes.setEnabled(true);
            chkBrowserNo.setEnabled(true);
         }
      }
      else if ( (eventSource == chkBrowserYes) || (eventSource == chkBrowserNo) ){
         mainAppletContext.fpNextItemPrompt.setDirection("right");
         mainAppletContext.fpNextItemPrompt.setLocation(chcBrowserName.getLocation().x - 30,
 chcBrowserName.getLocation().y);
         chcBrowserName.setEnabled(true);
      }
      else if (eventSource == chcBrowserName){
         if (! chcBrowserName.getSelectedItem().equals("Select One")){
            mainAppletContext.fpNextItemPrompt.setDirection("right");
            mainAppletContext.fpNextItemPrompt.setLocation(chkViewedWebPageYes.getLocation()
.x - 30, chkViewedWebPageYes.getLocation().y);
            chkViewedWebPageYes.setEnabled(true);
            chkViewedWebPageNo.setEnabled(true);
         }
      }
      else if ( (eventSource == chkViewedWebPageYes) || (eventSource == chkViewedWebPageNo)
){
         mainAppletContext.fpNextItemPrompt.setDirection("right");
         mainAppletContext.fpNextItemPrompt.setLocation(btnSubmit.getLocation().x -30, btnSu
```

```
bmit.getLocation().y);
            btnSubmit.setEnabled(true);
         }
      }

      public void keyPressed(KeyEvent ke){}
      public void keyReleased(KeyEvent ke){}
      public void keyTyped(KeyEvent ke){}

      public void windowActivated(WindowEvent we){}
      public void windowClosed(WindowEvent we){}
      public void windowClosing(WindowEvent we){
         setVisible(false);
      }
      public void windowDeactivated(WindowEvent we){}
      public void windowDeiconified(WindowEvent we){}
      public void windowIconified(WindowEvent we){}
      public void windowOpened(WindowEvent we){}


}
```

```
import java.awt.*;
import java.awt.event.*;
public class DartAppletMapOverviewWindow extends Frame implements WindowListener, ActionList
ener{
   DartApplet mainAppletContext;
   MouseOverButton btnOK;
   Image mapOverviewImage;
   DartAppletMapOverviewWindow(DartApplet appletContext){
      mainAppletContext = appletContext;
      setLayout(null);
      setBackground(Color.lightGray);
      setSize(300, 300);
      setLocation(100, 100);
      setResizable(false);
      setTitle("Map Overview");
      btnOK = new MouseOverButton("OK");
      btnOK.setSize(40, 20);
      btnOK.setLocation(130, 240);
      btnOK.setFont(mainAppletContext.regularScreenFont);
      btnOK.addActionListener(this);
      add(btnOK);
      addWindowListener(this);
   }

   public void setImage(Image mapOverviewImage){
      if (this.mapOverviewImage != null){
        this.mapOverviewImage.flush();
      }
      this.mapOverviewImage = mapOverviewImage;
      repaint();
   }

   public void paint(Graphics g){
      if (mapOverviewImage != null){
         g.drawImage(mapOverviewImage, 70, 60, null);
      }
      g.drawRect(70, 60, mapOverviewImage.getWidth(null), mapOverviewImage.getHeight(null));
   }

   public void actionPerformed(ActionEvent ae){
      setVisible(false);  // right now the only actionEvent will be the "OK" button being pr
essed
   }

   public void windowActivated(WindowEvent we){}
   public void windowClosed(WindowEvent we){}
   public void windowClosing(WindowEvent we){
      setVisible(false);
   }
   public void windowDeactivated(WindowEvent we){}
   public void windowDeiconified(WindowEvent we){}
   public void windowIconified(WindowEvent we){}
   public void windowOpened(WindowEvent we){}

}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class DartAppletLEAStatisticsWindow extends Frame implements WindowListener, ActionLi
stener{
   DartApplet mainAppletContext;
   MouseOverButton btnOK;
   Vector vtrLEARecords;
   DartAppletLEAStatisticsWindow(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setSize(320, 290);
      setLocation(100, 100);
      setResizable(false);
      setLayout(null);
      setTitle("Statistics");
      btnOK = new MouseOverButton("OK");
      btnOK.setFont(mainAppletContext.regularScreenFont);
      btnOK.addActionListener(this);
      btnOK.setSize(40, 20);
      btnOK.setLocation(120, 220);
      add(btnOK);
      addWindowListener(this);
   }

   public void show(){
      Object[] dataTransportArray = new Object[2];
      dataTransportArray[0] = "getLEAsForCDC";
      dataTransportArray[1] = mainAppletContext.scrCDCInfoScreen.currentRecord.username;
      dataTransportArray = mainAppletContext.doServletRequest(dataTransportArray);
      vtrLEARecords = (Vector)(dataTransportArray[0]);
      super.show();
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.screenTitleFont);
      g.drawString("There are currently " , 30, 70);
      g.setColor(Color.red);
      g.drawString("" + vtrLEARecords.size(), 120, 110);
      g.setColor(Color.black);
      g.drawString("LEAs for the state of", 20, 150);
      g.setColor(Color.red);
      g.drawString(mainAppletContext.scrCDCInfoScreen.currentRecord.state, 120, 190);
   }

   public void actionPerformed(ActionEvent ae){
      setVisible(false);  // right now the only action is the "OK" button being pressed
   }

   public void windowActivated(WindowEvent we){}
   public void windowClosed(WindowEvent we){}
   public void windowClosing(WindowEvent we){
      setVisible(false);
   }
   public void windowDeactivated(WindowEvent we){}
   public void windowDeiconified(WindowEvent we){}
   public void windowIconified(WindowEvent we){}
   public void windowOpened(WindowEvent we){}

}
```

```
import java.awt.*;

public class DartAppletCDPlannerScreen1 extends Panel{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   DartAppletChangeScreenButton backButton, forwardButton, homeButton;
   public DartAppletCDPlannerScreen1(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);

      backButton = new DartAppletChangeScreenButton(this, mainAppletContext.scrMainScreen, "
back");
      backButton.setSize(30, 30);
      backButton.setLocation(320, 540);
      add(backButton);

      forwardButton = new DartAppletChangeScreenButton(this, mainAppletContext.scrMainScreen
, "forward");
      forwardButton.setSize(30, 30);
      forwardButton.setLocation(360, 540);
      add(forwardButton);

      homeButton = new DartAppletChangeScreenButton(this, mainAppletContext.scrMainScreen, "
home");
      homeButton.setSize(30, 30);
      homeButton.setLocation(400, 540);
      add(homeButton);
   }
   public void paint(Graphics g){
      g.setFont(mainAppletContext.screenTitleFont);
      fm = g.getFontMetrics();

      tempString = "CD PLANNER";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }
}
```

```
import java.awt.*;
import java.awt.event.*;

public class DartAppletChangeScreenButton extends Canvas implements MouseListener, MouseMoti
onListener{
   Panel screenCurrentlyIn;
   Panel screenToGoTo;
   boolean blnMouseInside = false;
   boolean blnMouseInsidePreviously = false;
   String direction;
   Polygon plgSymbol;
   Image buffer;
   Graphics bufferG;
   public DartAppletChangeScreenButton(Panel screenCurrentlyIn, Panel screenToGoTo, String d
irection){
      this.screenCurrentlyIn = screenCurrentlyIn;
      this.screenToGoTo = screenToGoTo;
      this.direction = direction;
      addMouseListener(this);
      addMouseMotionListener(this);
   }
   public void paint(Graphics g){
      if (buffer == null){
         buffer = createImage(getSize().width, getSize().height);
         bufferG = buffer.getGraphics();
      }
      if (plgSymbol == null){
         plgSymbol = new Polygon();
         if (direction.equals("back")){
            plgSymbol.addPoint(0, (int)(getSize().height/2.0 + 0.5));
            plgSymbol.addPoint( (int)(0.4 * getSize().width + 0.5), (int)(0.1 * getSize().he
ight + 0.5));
            plgSymbol.addPoint( (int)(0.4 * getSize().width + 0.5), (int)(0.3 * getSize().he
ight + 0.5));
            plgSymbol.addPoint(getSize().width -1, (int)(0.3 * getSize().height + 0.5));
            plgSymbol.addPoint(getSize().width -1, (int)(0.7 * getSize().height + 0.5));
            plgSymbol.addPoint( (int)(0.4 * getSize().width + 0.5), (int)(0.7 * getSize().he
ight + 0.5));
            plgSymbol.addPoint( (int)(0.4 * getSize().width + 0.5), (int)(0.9 * getSize().he
ight + 0.5));
         }
         else if (direction.equals("forward")){
            plgSymbol.addPoint(0, (int)(0.7 * getSize().height + 0.5));
            plgSymbol.addPoint(0, (int)(0.3 * getSize().height + 0.5));
            plgSymbol.addPoint( (int)(0.6 * getSize().width + 0.5), (int)(0.3 * getSize().he
ight + 0.5));
            plgSymbol.addPoint( (int)(0.6 * getSize().width + 0.5), (int)(0.1 * getSize().he
ight + 0.5));
            plgSymbol.addPoint(getSize().width -1, (int)(0.5 * getSize().height + 0.5));
            plgSymbol.addPoint( (int)(0.6 * getSize().width + 0.5), (int)(0.9 * getSize().he
ight + 0.5));
            plgSymbol.addPoint( (int)(0.6 * getSize().width + 0.5), (int)(0.7 * getSize().he
ight + 0.5));
         }
         else if (direction.equals("home")){
            plgSymbol.addPoint( (int)(0.20 * getSize().width + 0.5), getSize().height -1);
            plgSymbol.addPoint( (int)(0.20 * getSize().width + 0.5), (int)(0.6 * getSize().h
eight + 0.5));
            plgSymbol.addPoint(0, (int)(0.6 * getSize().height + 0.5));
            plgSymbol.addPoint( (int)(0.5 * getSize().width + 0.5), 0);
            plgSymbol.addPoint(getSize().width -1, (int)(0.6 * getSize().height + 0.5));
            plgSymbol.addPoint( (int)(0.8 * getSize().width + 0.5), (int)(0.6 * getSize().he
ight + 0.5));
            plgSymbol.addPoint( (int)(0.8 * getSize().width + 0.5), getSize().height -1);
         }
```

```
      }

      if (blnMouseInside){
         bufferG.setColor(Color.blue);
      }
      else{
         bufferG.setColor(Color.black);
      }
      bufferG.fillPolygon(plgSymbol);
      g.drawImage(buffer, 0, 0, null);
   }
   public void update(Graphics g){
      paint(g);
   }

   public void mouseEntered(MouseEvent me){}
   public void mouseExited(MouseEvent me){
      blnMouseInside = false;
      repaint();
   }
   public void mousePressed(MouseEvent me){}
   public void mouseReleased(MouseEvent me){}
   public void mouseClicked(MouseEvent me){
      if (blnMouseInside){
         screenCurrentlyIn.setVisible(false);
         screenToGoTo.setVisible(true);
      }
   }
   public void mouseDragged(MouseEvent me){}
   public void mouseMoved(MouseEvent me){
     blnMouseInsidePreviously = blnMouseInside;
     if (plgSymbol.contains(me.getX(), me.getY())){
        blnMouseInside = true;
     }
     else{
        blnMouseInside = false;
     }
     if ( blnMouseInsidePreviously != blnMouseInside){
        repaint();
     }
   }
}
```

C:\InetPub\wwwroot\DartApplet\DartAppletDMISpatialIntegratorScreen1.java 07/26/00 10:11:AM

```
import java.awt.*;

public class DartAppletDMISpatialIntegratorScreen1 extends Panel{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   DartAppletChangeScreenButton backButton, forwardButton, homeButton;
   public DartAppletDMISpatialIntegratorScreen1(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);

      backButton = new DartAppletChangeScreenButton(this, mainAppletContext.scrMainScreen, "
back");
      backButton.setSize(30, 30);
      backButton.setLocation(320, 540);
      add(backButton);

      forwardButton = new DartAppletChangeScreenButton(this, mainAppletContext.scrMainScreen
, "forward");
      forwardButton.setSize(30, 30);
      forwardButton.setLocation(360, 540);
      add(forwardButton);

      homeButton = new DartAppletChangeScreenButton(this, mainAppletContext.scrMainScreen, "
home");
      homeButton.setSize(30, 30);
      homeButton.setLocation(400, 540);
      add(homeButton);
   }
   public void paint(Graphics g){
      g.setFont(mainAppletContext.screenTitleFont);
      fm = g.getFontMetrics();

      tempString = "DMI SPATIAL INTEGRATOR";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }
}
```

```
import java.awt.*;
import java.awt.event.*;

public class DartAppletLEAITCensusWindow extends Frame implements KeyListener, ActionListene
r, WindowListener, ItemListener{
   DartApplet mainAppletContext;
   DartAppletLEAITCensusWindow(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setSize(300, 300);
      setLocation(100, 100);
      setResizable(false);
      setTitle("IT Census");
      addWindowListener(this);
   }

   public void windowActivated(WindowEvent we){}
   public void windowClosed(WindowEvent we){}
   public void windowClosing(WindowEvent we){
      setVisible(false);
   }
   public void windowDeactivated(WindowEvent we){}
   public void windowDeiconified(WindowEvent we){}
   public void windowIconified(WindowEvent we){}
   public void windowOpened(WindowEvent we){}

}
```

```
import java.awt.*;
import java.awt.event.*;

public class DartAppletLoginFailureScreen extends Panel implements ActionListener{
   DartApplet mainAppletContext;
   FontMetrics fm;
   String tempString;
   Panel screenToGoBackTo;
   MouseOverButton btnOK;

   public DartAppletLoginFailureScreen(DartApplet appletContext){
      mainAppletContext = appletContext;
      setBackground(Color.lightGray);
      setLayout(null);

      btnOK = new MouseOverButton("OK");
      btnOK.setSize(50, 20);
      btnOK.setLocation(100, 220);
      btnOK.addActionListener(this);
      add(btnOK);
   }

   public void setVisible(boolean visible){
      super.setVisible(visible);
      if (visible){
         btnOK.requestFocus();
      }
   }

   public void setScreenToGoBackTo(Panel screenToGoBackTo){
      this.screenToGoBackTo = screenToGoBackTo;
   }

   public void paint(Graphics g){
      g.setFont(mainAppletContext.smallerScreenTitleFont);
      fm = g.getFontMetrics();

      tempString = "Login Failure";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 40);

      g.setFont(mainAppletContext.regularScreenFont);
      fm = g.getFontMetrics();
      tempString = "Please check your";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 105);
      tempString = "username and password";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 135);
      tempString = "and try again.";
      g.drawString(tempString, (int)(getSize().width/2.0 - fm.stringWidth(tempString)/2.0 +
0.5), 165);

      mainAppletContext.drawWindowBorder(g, getSize().width, getSize().height);
   }

   public void btnOKClicked(){
      setVisible(false);
      screenToGoBackTo.setVisible(true);
   }

   public void actionPerformed(ActionEvent ae){
      String ac = ae.getActionCommand();
      if (ac.equals("OK")){
         btnOKClicked();
```

```
        }
    }

}
```

```
        }
    }

}
```

```
import java.awt.*;
public class AutoLabel extends Label{
   public AutoLabel(String title, Font labelFont){
      setFont(labelFont);
      setText(title);
      FontMetrics fm = getFontMetrics(labelFont);
      int width = fm.stringWidth(title);
      setSize(width + 3, fm.getHeight());
   }
}
```

Appendix E – Continued

DART ArcView Server/RPC Scripts

ArcView® Avenue™ Code

The following pages provide print outs of the individual Avenue scripts used to maintain the Remote Procedure Call (RPC) interface to the DART JAVA™ Applet.

_Startup Script

```
RPCServer.Start(0x40000001,1)
```

_Stop Script

```
RPCServer.Stop
```

CheckForRequest Script

```
thePath = "c:\inetpub\wwwroot\DartApplet\ImageRequests".AsFileName
theFiles = thePath.ReadFiles("*.txt")
if (theFiles.IsEmpty.Not) then
  for each f in theFiles
    av.Run("ProcessFile",f)
    File.Delete(f)
  end
end
av.DelayedRun("CheckforRequest","",1)
```

CreateDartView Script

```
ViewType = SELF.Get(0).UCase
aName = SELF.Get(1)
aPoint = SELF.Get(2)
theState = SELF.Get(3)
```

```
JurType = SELF.Get(4)
JurName = SELF.Get(5)
baseFileName = SELF.Get(6)
ImageFN = (baseFileName+".jpg").AsFileName
FinishedFN = (baseFileName+".finished").AsFileName
StateName = av.Run("LookupStateName",theState)
'theView = av.GetProject.FindDoc("USA").clone
theView = av.GetProject.FindDoc("USA")
theGL = theView.GetGraphics
theGL.Empty
'---------------------------------------
'Draw CDC Data and Text
'---------------------------------------
CDCTheme = theView.FindTheme("CDC Data")
CDCFTab = CDCTheme.GetFTab
CDCFTab.Refresh
'theQuery = "[Name] ="++theName.Quote
'theFTab.SetDefinition(theQuery)
'theFTab.UpdateDefBitmap
CDCSym = TextSymbol.Make
CDCSym.SetSize(16)
CDCSym.SetColor(Color.GetBlack)
theFont = Font.Make(CDCSym.GetFont.GetFamily,"Bold")
CDCSym.SetFont(theFont)
LEASym = TextSymbol.Make
LEASym.SetSize(15)
LEASym.SetColor(Color.GetBlack)
theFont = Font.Make(LEASym.GetFont.GetFamily,"Bold")
LEASym.SetFont(theFont)
labelField = CDCTheme.GetLabelField
for each rec in CDCFTab
  theValue = CDCFTab.ReturnValue(labelField,rec).Trim
  thePoint = CDCFTab.GetLabelPoint(rec)
  if (theValue <> "") then
    gt = GraphicText.Make(theValue,thePoint)
    gt.SetSymbol(CDCSym)
    theGL.AddBatch(gt)
  end
end
theGL.EndBatch
'---------------------------------------
'Draw LEA Data and Text
'---------------------------------------
LEAName = aName
LEATheme = theView.FindTheme("LEA Data")
LEATab = LEATheme.GetFTab
```

```
LEATab.GetSelection.ClearAll
LEATab.UpdateSelection
LEATab.Refresh
theQuery = "[Name] ="++LEAName.Quote
if (ViewType <> "CDCVIEW") then
 if (ViewType = "LEAVIEW") then
  LEATab.SetDefinition(theQuery)
  LEATab.UpdateDefBitmap
  found = (LEATab.GetDefBitmap.Count > 0)
 elseif (ViewType = "LEAVIEWWITHLEAS") then
  LEASel = LEATab.GetSelection
  LEATab.Query(theQuery,LEASel,#VTAB_SELTYPE_NEW)
  LEATab.UpdateSelection
  found = (LEASel.Count > 0)
 else
  found = True
  'Draw all LEAs (CDCVIEWWITHLEAS)
 end
 labelField = LEATheme.GetLabelField
 if (found.Not and (aPoint.GetX <> 0)) then
  gt = GraphicText.Make(LEAName,aPoint)
  gt.SetSymbol(LEASym)
  theGL.AddBatch(gt)
 end
 for each rec in LEATab
  theValue = LEATab.ReturnValue(labelField,rec).Trim
  thePoint = LEATab.GetLabelPoint(rec)
  if (theValue <> "") then
   gt = GraphicText.Make(theValue,thePoint)
   gt.SetSymbol(LEASym)
   theGL.AddBatch(gt)
  end
 end
 theGL.EndBatch
end
'---------------------------------
'Get extent and create view
'---------------------------------
if (StateName <> nil) then
 theExt = av.Run("GetJurisdictionExtent",{StateName,JurType,JurName})
 if (theExt <> nil) then
  av.Run("SetViewExtent",{theExt, theView})
 else
  ImageFN = nil
 end
else
```

```
 ImageFN = nil
end
theView.GetWin.Open
theView.GetDisplay.Flush
av.Run("ExportImage",{theView,ImageFN,FinishedFN})
```

CreateOverview Script

```
ViewType = SELF.Get(0).UCase
aName = SELF.Get(1)
aPoint = SELF.Get(2)
theState = SELF.Get(3)
JurType = SELF.Get(4)
JurName = SELF.Get(5)
baseFileName = SELF.Get(6)
ImageFN = (baseFileName+".jpg").AsFileName
FinishedFN = (baseFileName+".finished").AsFileName
StateName = av.Run("LookupStateName",theState)
theView = av.GetProject.FindDoc("OverView")
theGL = theView.GetGraphics
theGL.Empty
'-------------------------------------
' Draw CDC Data and Text
'-------------------------------------
'CDCTheme = theView.FindTheme("CDC Data")
'CDCFTab = CDCTheme.GetFTab
'CDCFTab.Refresh
''theQuery = "[Name] ="++theName.Quote
''theFTab.SetDefinition(theQuery)
''theFTab.UpdateDefBitmap
'CDCSym = TextSymbol.Make
'CDCSym.SetSize(16)
'CDCSym.SetColor(Color.GetBlack)
'theFont = Font.Make(CDCSym.GetFont.GetFamily,"Bold")
'CDCSym.SetFont(theFont)
'LEASym = TextSymbol.Make
'LEASym.SetSize(15)
'LEASym.SetColor(Color.GetBlack)
'theFont = Font.Make(LEASym.GetFont.GetFamily,"Bold")
'LEASym.SetFont(theFont)
```

```
'labelField = CDCTheme.GetLabelField
'for each rec in CDCFTab
' theValue = CDCFTab.ReturnValue(labelField,rec).Trim
' thePoint = CDCFTab.GetLabelPoint(rec)
' if (theValue <> "") then
'   gt = GraphicText.Make(theValue,thePoint)
'   gt.SetSymbol(CDCSym)
'   theGL.AddBatch(gt)
' end
'end
'theGL.EndBatch
'-------------------------------
'Get extent and create view
'-------------------------------
if (StateName <> nil) then
 ViewExt = av.Run("GetJurisdictionExtent",{StateName,JurType,JurName})
 OverViewExt = av.Run("GetJurisdictionExtent",{StateName,"State",""})
 if ((OverViewExt = nil) or (ViewExt = nil)) then
  ImageFN = nil
 else
  theView.GetDisplay.SetExtent(OverViewExt.Scale(1.1))
  theOutline = GraphicShape.Make(ViewExt)
  theOutline.GetSymbol.SetOLColor(Color.GetRed)
  theOutline.GetSymbol.SetOLWidth(2)
  theView.GetGraphics.Add(theOutline)
 end
else
 ImageFN = nil
end
theView.GetWin.Open
theView.GetDisplay.Flush
av.Run("ExportImage",{theView,ImageFN,FinishedFN})
```

CreateTutorialView Script

```
bitString = SELF.Get(1)
baseFileName = SELF.Get(2)
ImageFN = (baseFileName+".jpg").AsFileName
FinishedFN = (baseFileName+".finished").AsFileName
StatesOn = (bitString.Left(1) = "1")
```

```
CountiesOn = (bitString.Middle(1,1) = "1")
RoadsOn = (bitString.Middle(2,1) = "1")
RiversOn = (bitString.Middle(3,1) = "1")
theView = av.GetProject.FindDoc("Tutorial")
theStates = theView.FindTheme("States")
theCnty = theView.FindTheme("Counties")
theRoads = theView.FindTheme("Roads")
theRivers = theView.FindTheme("Rivers")
theStates.SetVisible(StatesOn)
theCnty.SetVisible(CountiesOn)
theRoads.SetVisible(RoadsOn)
theRivers.SetVisible(RiversOn)
theView.GetWin.Open
theView.GetDisplay.Flush
av.Run("ExportImage",{theView,ImageFN,FinishedFN})
```

ExportImage Script

```
theView = SELF.Get(0)
'theView.GetWin.Open
theImageFN = SELF.Get(1)
FinishedFileName = SELF.Get(2)
if (theImageFN <> nil) then
  theView.ExportToFile(theImageFN,"JPEG",{96,100})
end
f = LineFile.Make(FinishedFileName, #FILE_PERM_WRITE)
f.WriteElt("file is finished")
f.close
'av.GetProject.RemoveDoc(theView)
```

GetCDCExtent Script

```
theState = SELF
theView = av.GetProject.FindDoc("ProcessView")
```

```
stateTheme = theView.FindTheme("States")
stateTab = stateTheme.GetFTab
theExt = Nil
StateName = av.Run("LookupStateName",theState)
if (StateName <> nil) then
 queryStr = "[state_name] = "++StateName.Quote
 stateSel = stateTab.GetSelection
 stateTab.Query(queryStr,stateSel,#VTAB_SELTYPE_NEW)
 stateTab.UpdateSelection
 if (stateTab.GetNumSelRecords <> 1) then
  return nil
 end
 theExt = stateTheme.GetSelectedExtent
end
return theExt
```

GetJurisdictionExtent Script

```
StateName = SELF.Get(0)
JurType = SELF.Get(1)
JurName = SELF.Get(2)
ProcessView = av.GetProject.FindDoc("ProcessView")
JurTheme = ProcessView.FindTheme(JurType)
JurTab = JurTheme.GetFTab
JurTab.GetSelection.ClearAll
JurTab.UpdateSelection
theExt = nil
if (JurTheme = nil) then
 return theExt
end
if (JurType = "State") then
 queryStr = "[state_name] = "++StateName.Quote
 theSel = JurTab.GetSelection
 JurTab.Query(queryStr,theSel,#VTAB_SELTYPE_NEW)
 JurTab.UpdateSelection
elseif (JurType = "County") then
 queryStr = "([state_name] ="++StateName.Quote+") and ([name] =
"++JurName.Quote+")"
 theSel = JurTab.GetSelection
 JurTab.Query(queryStr,theSel,#VTAB_SELTYPE_NEW)
```

```
  JurTab.UpdateSelection
elseif (JurType = "Urban") then
  queryStr = "([place_name] ="++JurName.Quote+")"
  theSel = JurTab.GetSelection
  JurTab.Query(queryStr,theSel,#VTAB_SELTYPE_NEW)
  JurTab.UpdateSelection
else
  JurTab.GetSelection.ClearAll
  JurTab.UpdateSelection
end
if (JurTab.GetNumSelRecords = 1) then
  theExt = JurTheme.GetSelectedExtent
end
return theExt
```

LookupStateName Script

```
theState = SELF
theD = Dictionary.Make(50)
theD.Add("AL","Alabama")
theD.Add("AK","Alaska")
theD.Add("AZ","Arizona")
theD.Add("AR","Arkansas")
theD.Add("AS","American Samoa")
theD.Add("CA","California")
theD.Add("CO","Colorado")
theD.Add("CT","Connecticut")
theD.Add("DE","Delaware")
theD.Add("DC","District of Columbia")
theD.Add("FM","Federated States Of Micronesia")
theD.Add("FL","Florida")
theD.Add("GA","Georgia")
theD.Add("GU","Guam")
theD.Add("HI","Hawaii")
theD.Add("ID","Idaho")
theD.Add("IL","Illinois")
theD.Add("IN","Indiana")
theD.Add("IA","Iowa")
theD.Add("KS","Kansas")
theD.Add("KY","Kentucky")
```

```
theD.Add("LA","Louisiana")
theD.Add("ME","Maine")
theD.Add("MH","Marshall Islands")
theD.Add("MD","Maryland")
theD.Add("MA","Massachusetts")
theD.Add("MI","Michigan")
theD.Add("MN","Minnesota")
theD.Add("MS","Mississippi")
theD.Add("MO","Missouri")
theD.Add("MT","Montana")
theD.Add("NE","Nebraska")
theD.Add("NV","Nevada")
theD.Add("NH","New Hampshire")
theD.Add("NJ","New Jersey")
theD.Add("NM","New Mexico")
theD.Add("NY","New York")
theD.Add("NC","North Carolina")
theD.Add("ND","North Dakota")
theD.Add("MP","Northern Mariana Islands")
theD.Add("OH","Ohio")
theD.Add("OK","Oklahoma")
theD.Add("OR","Oregon")
theD.Add("PW","Palau")
theD.Add("PA","Pennsylvania")
theD.Add("PR","Puerto Rico")
theD.Add("RI","Rhode Island")
theD.Add("SC","South Carolina")
theD.Add("SD","South Dakota")
theD.Add("TN","Tennessee")
theD.Add("TX","Texas")
theD.Add("UT","Utah")
theD.Add("VT","Vermont")
theD.Add("VI","Virgin Islands")
theD.Add("VA","Virginia")
theD.Add("WA","Washington")
theD.Add("WV","West Virginia")
theD.Add("WI","Wisconsin")
theD.Add("WY","Wyoming")
return theD.Get(theState)
```

ProcessFile Script

```
theFile = SELF
theView = av.GetProject.FindDoc("USA")
theGL = theView.GetGraphics
fp = LineFile.Make(theFile,#FILE_PERM_READ)
ViewType = fp.ReadElt
if (ViewType = "CDCVIEW") then
 theState = fp.ReadElt
 theCoords = fp.ReadElt
 ImageName = fp.ReadElt
 FinishedFileName = fp.ReadElt
 theLat = theCoords.AsTokens(" ,"+tab).Get(0).AsNumber
 theLon = theCoords.AsTokens(" ,"+tab).Get(1).AsNumber
 theLoc = theLon@theLat
 if (theGL.IsEmpty)  then
  theGL.Add(GraphicShape.Make(theLoc))
 else
  theGraphic = theGL.Get(0)
  theGraphic.Invalidate
  theGraphic.SetShape(theLoc)
  theGraphic.Invalidate
 end
 theExt = av.Run("GetCDCExtent",theState)
 av.Run("View_Image.Make",{theExt,ImageName.AsFileName})
end
fp.Close
f = LineFile.Make(FinishedFileName.AsFileName, #FILE_PERM_WRITE)
f.WriteElt("file is finished")
f.close
av.PurgeObjects
```

ProcessRequest Script

```
if (SELF.Get(0).UCase = "TUTORIALVIEW") then
 av.Run("CreateTutorialView",SELF)
elseif (SELF.Get(0).UCase = "LEAOVERVIEW") then
 av.Run("CreateOverView",SELF)
else
```

```
UrbanTheme.SelectByRect(theExt,#VTAB_SELTYPE_NEW)
UrbanTab = UrbanTheme.GetFTab
UrbanTab.UpdateSelection
labelField = UrbanTheme.GetLabelField
for each rec in UrbanTab.GetSelection
  theValue = UrbanTab.ReturnValue(labelField,rec).Trim
  thePoint = UrbanTab.GetLabelPoint(rec)
  if (theValue <> "") then
    gt = GraphicText.Make(theValue,thePoint)
    gt.SetSymbol(citySym)
    theGL.AddBatch(gt)
  end
 end
 theGL.EndBatch
end
```

StartRPC Script

```
'av.Run("_Stop", "")
av.DelayedRun("_Startup", "", 5)
```

TestExport Script

```
theExt = av.GetActiveDoc.GetDisplay.ReturnUserRect
theFileName = av.GetProject.MakeFileName("image","jpg")
av.Run("View_Image.Make",{theExt,theFileName})
```

```
 av.Run("CreateDartView",SELF)
end
```

SetViewExtent Script

```
theExt = SELF.Get(0)
theView = Self.Get(1)
ProcessView = av.GetProject.FindDoc("ProcessView")
theView.GetWin.Open
theGL = theView.GetGraphics
theView.GetDisplay.SetExtent(theExt.Scale(1.1))
citySym = TextSymbol.Make
citySym.SetSize(9)
countySym = TextSymbol.Make
countySym.SetSize(14)
theFont = countySym.GetFont
theFamily = theFont.GetFamily
newFont = Font.Make(theFamily,"Italic")
countySym.SetFont(newFont)
'Add text for Counties if scale < 2000000
if (theView.ReturnScale <= 2000000) then
  Ctytheme = ProcessView.FindTheme("County")
  CtyTheme.SelectByRect(theExt,#VTAB_SELTYPE_NEW)
  CtyTab = CtyTheme.GetFTab
  CtyTab.UpdateSelection
  labelField = CtyTheme.GetLabelField
  for each rec in CtyTab.GetSelection
    theValue = CtyTab.ReturnValue(labelField,rec).Trim
    thePoint = CtyTab.GetLabelPoint(rec)
    if (theValue <> "") then
      gt = GraphicText.Make(theValue,thePoint)
      gt.SetSymbol(countySym)
      theGL.AddBatch(gt)
    end
  end
  theGL.EndBatch
end
'Add text for urban areas if scale < 1000000
if (theView.ReturnScale <= 1000000) then
  Urbantheme = ProcessView.FindTheme("Urban")
```

TestProcessRequest Script

```
'av.Run("ProcessRequest",{"John
Breckenridge",1,"MS","State","Simpson","e:\temp\test1"})
av.Run("ProcessRequest",{"TutorialView","1011","e:\temp\test1"})
```